# Второй курс, осенний семестр 2022/23
# Практика по алгоритмам #11

## Мощь Фурье

## 9 декабря

Собрано 9 декабря 2022 г. в 11:58

---

## Содержание

# Мощь Фурье

1. **Возведение в степень**

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

2. **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 $\mathcal{O}(k|t| \log |t|)$.

   Посчитайте число вызовов FFT. (*) Как ускорить в 3 раза?

3. **Поиск с ошибками и шаблоном**

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

4. **Поиск подстроки в строке**

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

5. **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 $j - i = k - j$.

6. **3-SUM**

   Формулировка 3-SUM: дан массив $A$, выбрать $a, b, c \in A$: $a + b + c = 0$. В общем виде задача не решается за $\mathcal{O}(n^{2-\varepsilon})$, но, если $A \subseteq \mathbb{Z} \cap [-C, C]$, есть простое решение за $\mathcal{O}(C \log C)$.

7. **Уравнение**

   Даны $n$ и $m$. Найти число троек $(x, y, z)$: $x^n + y^n \equiv z^n \bmod m$. $m \leqslant 10^6$.
   (*) Ускорьте предподсчёт, чтобы при $n = m$ работал за $\Theta(n)$.

8. **Цепочка умножений**

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

9. **Одно `FFT` через несколько `FFT`**

   Сведите вычисление `FFT` последовательности размера $pn$ к $p$ вычислениям `FFT` от последовательностей размера $n$ и $\mathcal{O}(p^2 n)$ дополнительных операций.
   Какие точки выбрать? Какую группу взять?

10. **`FFT` по простому модулю**

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

11. **`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 $\mathbb{Z}/p\mathbb{Z}$.

## 12. 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;
```

## 13. (*) Том-Кук

Вашему вниманию представляется простой алгоритм умножения многочленов за $\mathcal{O}(n^{1+\varepsilon})$. Рассмотрим его на примере $k = 3$. Берём исходные многочлены $A(x)$ и $B(x)$ степени $n$, делим оба по 3 отрезка длины $\frac{n}{3}$, получили многочлены $A_1, A_2, A_3, B_1, B_2, B_3$.

Запишем $A(t) = A_1(x)t^2 + A_2(x)t + A_3(x)$, это верное равенство при $t = x^{n/3}$.
Аналогично $B(t) = B_1(x)t^2 + B_2(x)t + B_3(x)$. Наша задача посчитать $C(t) = A(t)B(t)$, это многочлен степени 4 от $t$. Посчитаем его значения в 5 точках $(t = 0, 1, 2, 3, 4)$ и интерполируем.

a) Додумайте решения. Должно получить 5 рекурсивных вызовов от $\frac{n}{3}$.
b) Обобщите на $\forall k$, оцените время работы от $k$. Должно получиться $n^{f(k)}$: $f(k) \xrightarrow{k\to+\infty} 1$.

## 14. (*) Задача о рюкзаке

Даны $n$ предметов и запросы «можно ли набрать вес $w_i$, используя только предметы с номерами от $l_i$ до $r_i$». При этом все $w_i \leqslant s$. Сделайте предподсчёт за $\mathcal{O}(ns \log s)$ так, чтобы на запрос можно было бы в online ответить за $\mathcal{O}(s \log s \log n)$.

P.S. Есть решение динамикой за $\mathcal{O}(ns)$.

15. (*) **Пентагональная теорема Эйлера**

Сама теорема заключается в том, что $Q(x) = \prod\limits_{k=1}^{\infty}(1-x^k) = \sum\limits_{q=-\infty}^{\infty}(-1)^q x^{(3q^2+q)/2}$. Рассмотрим $P(x) = \sum_n p_n x^n$, где $p_n$ – число разбиений числа $n$ на неубывающие слагаемые. Заметим:

$$P(x) = \prod_{k=1}^{\infty}(1 + x^k + x^{2k} + \dots) = \prod_{k=1}^{\infty}\tfrac{1}{1-x^k}$$

Используя эти знания найдите количество разложений числа $n \leqslant 2^{18}$ на неубывающие слагаемые по модулю $3 \cdot 2^{18}+1$ (простое).

a) За $\mathcal{O}(n^{3/2})$ в лоб.
b) За $\mathcal{O}(n \log^2 n)$ и $\mathcal{O}(n \log n)$, используя мозг и FFT.
c) Предложите, как найти само число, а не только остаток от деления.
За $\mathcal{O}(mn \log n + m^2)$, где $m$ – длина ответа.

# Разбор задач практики

1. **Возведение в степень**

   Возведим 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 $n = 10^6$».

2. **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.

3. **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.

4. **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!

5. **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 $\sum b_{2i} a_i - a_i^2$.

6. **3-SUM**

   Рассмотрим $A(x) = \sum_i x^{a_i}$, нам нужно посмотреть коэффициент при $x^S$ у $A^3(x)$.

7. **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 $b = a^2, res = \sum b_i a_i$.

8. **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 $\mathcal{O}(\frac{k}{2}FFT(2n) + \frac{k}{4}FFT(4n) + ... + FFT(kn)) = \mathcal{O}(nk \log(nk))$.

9. **Одно FFT через несколько FFT**

   Для простоты распишем для $p = 3$, дальше легко обобщить. $P(x) = P_0(x^3)+xP_1(x^3)+x^2P(x^3)$. Посчитали в $n$ точках все $P_i$, потом в $3n$ точках пересчитаем $P$ по формуле выше.
   Заметим, $T(n) = 3T(\frac{n}{3}) + 3n$, а в общем случае $T(n) = kT(\frac{n}{k}) + kn = kn \log_k n$.

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. **Inplace FFT**

    `rev[rev[i]] = i` (то есть `rev` — инволюция), так что все циклы в перестановке имеют длину не более 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;
```

13. **(*) Том-Кук**

    Для произвольного $k$ имеем $\deg C = 2k \Rightarrow$ считать нужно в $2k+1$ точке.
    $\forall t$ $A(t), B(t)$ — многочлены от $x$. Сами $A(t)$ и $B(t)$ вычисляются за $\mathcal{O}(\frac{n}{k}k)$, а $C(t) = A(t)B(t)$ вычисляется одним рекурсивном вызовом умножения.

    *Интерполяция:* $C(t)$ — линейные комбинации над $C_0(x), C_1(x), \dots, C_{2k}(x) \Rightarrow$ решаем за $\mathcal{O}(k^3)$ линейную систему уравнений $(2k+1) \times (2k+1)$, получаем решение в виде $C_i(x) = \sum_j \alpha_{ij} C(j)$, $\forall i$ вычисляем $C_i(x)$ за $\frac{n}{k}k$.

*Итого:* $T(n) = nk + (2k{+}1)T(\frac{n}{k}) + k^3 + nk$.
$\forall$ константы $k$ получили $T(n) = \Theta((2k{+}1)^{\log_k n}) = \Theta(n^{\log_k(2k+1)})$.
Для $k{=}3$ имеем $n^{1.465}$, для $k{=}4$ имеем $n^{1.4037}$.

14. **(*) Задача о рюкзаке**

Есть рюкзаки $A$ и $B$, объединим их в рюкзак $C$: $C[i] = \vee(A[j] \wedge B[i-j])$. Можно делать это с `FFT` за $s \log s$. После этого просто построим ДО на предметах, в вершине рюкзак на отрезке предметов. Запрос разбивается на $\log n$ рюкзаков, которые надо объединить.

15. **(*) Пентагональная теорема Эйлера**

Нам нужно посчитать $P(x) = \frac{1}{Q(x)}$, мы знаем $Q$, осталось обратить его за $\mathcal{O}(n \log n)$.
*Обращение:* пусть мы уже нашли первых $k$ цифр $\frac{1}{Q(x)}$, обозначим их за $I_k$, $degI_k = k - 1$.

$$I_k(x)Q_k(x) = 1 + x^k R_k(x)$$

Здесь $Q_k(x)$ – младшие $k$ коэффициентов многочлена $Q$.
Будем искать $I_{2k}(x) = I_k(x) + x^k T(x)$: $I_{2k}(x)Q(x) = 1 + x^k R_k(x) + x^k T(x)Q(x)$.
Теперь нужно, чтобы $R_k(x) + T(x)Q(x) = 1 + x^k R_{2k}(x)$, получаем $T(x) = -R_k(x)I_k(x)$.

$$I_{2k}(x) = I_k(x) - x^k R_k(x)I_k(x) = I_k(x)(1 - x^k R_k(x)) = I_k(x)(2 - I_k(x)Q_k(x))$$

Итого за два умножения многочленов длины $k$ мы перешли к первым $2k$ коэффициентам.
Оценка времени работы: $\texttt{FFT}(1) + \texttt{FFT}(2) + \texttt{FFT}(4) + \texttt{FFT}(8) + \cdots = \mathcal{O}(n \log n)$.

# Домашнее задание

## 3.1. Обязательная часть

1. (4) **Динамика по дереву**

   Сколько способов вырезать из полного бинарного дерева глубины $k$ поддерево размера $s$, содержащее корень исходного? Где здесь `FFT`?

2. (3) **Обратное по модулю**

   Найти к числу $x$ $(0 \leqslant x < m)$ обратное по модулю $m$ за $\mathcal{O}(\log^2 m)$. Длиннное $m$.

3. (2) **Циклические сдвиги**

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

4. (3) **Поиск подкартинки**

   Даны две картины, заданные 256 оттенками серого. То есть даны матрицы целых чисел $A$ и $B$. $A$ по обоим размерам больше $B$. Найти такое наложение матрицы $B$ на $A$, что суммарное квадратичное отклонение цветов минимально. То есть, найти такие $i$, $j$:

   $$\sum_{x,y}(B[x,y] - A[x+i, y+j])^2 \to \min$$

5. (4) **AVL деревья**

   Дано $n < 2^{16} = N = 2^H$. И число $h \leqslant 16 = H$.
   Найти количество AVL деревьев глубины $h$ из $n$ вершин по модулю $3 \cdot 2^{18}+1$.

   a) (2) Используя вещественное `FFT`, за $\mathcal{O}(NH^2)$.
   b) (2) Используя `FFT` по простому модулю, за $\mathcal{O}(NH^2)$.
   c) (+1) $\mathcal{O}(NH)$.

## 3.2. Дополнительная часть

1. (3) Перевод из 10-ой системы счисления в 2-ую быстрее квадрата.
2. (5) Интерполяция в произвольных точках быстрее квадрата.
3. (4) С помощью `FFT` за $\mathcal{O}^*(2^n)$ проверьте, можно ли вершины неорграфа покрасить в $k$ цветов.
4. (4) Количество счастливых билетов из $2n$ цифр по модулю $m$ за $\mathcal{O}(n \log n)$.
5. (4) **Поиск шаблона в шаблоне**

   Даны строка и текст. Оба могут содержать вопросы.
   Найти точное совпадение за $\mathcal{O}(1)$ вызовов Фурье. Алфавит – **не** константа!