Первый курс, весенний семестр
Лекции от Ромы Андреева
FFT, Длинка

FFT

1. Схема для умножения чисел через интерполяцию и экстраполяцию. Биекция n коэффициентов и значений в n точках.
2. Экстраполяция за $\mathcal{O}(n^2)$. Интерполяция по Лагранжу за $\mathcal{O}(n^2)$.
3. Прямое преобразование Фурье в комплексных числах рекурсивно. Написать псевдокод `FFT(n, a)`. $n = 2^k$.
4. Обратное Фурье: w^{-1}, reverse. Доказательство обоих методов.
5. Нерекурсивное Фурье. Reverse bits. Написать псевдокод `FFT(n, a)`. $n = 2^k$.
6. Выбор системы счисления для Фурье. Проблема с A^5: если число было длины n, то коэффициенты будут порядка n^4, типа `long double` не хватит. Выбор системы счисления для чисел длины n: $nb^2 \le 10^{18} \Rightarrow b = \sqrt{\frac{10^{18}}{n}} \Rightarrow b = 10^6$ при типе `long double`.
7. Фурье по простому модулю. Мы хотим посчитать значение во всех точках по модулю p. Их $p-1$, и они образуют циклическую группу. Найдём первообразную, ищем её в лоб: перебираем от 2 и проверяем за линию. Теперь пока размер группа делится на два, можно сделать рекурсивный переход и стандартный "разделяй и властвуй". Пример хорошего p: $2^{18}3 + 1$

Основы длинки

1. **Хранение в десятичной системе счисления.**

```
vector<int> a; // экономный по памяти, промышленный
int a[N]; // самый короткий код
int n, a[N]; // короткий и быстрый код
```

2. **Сложение, вычитание, сравнение, домножение и деление на короткое.**

```
void add( int *a, int *b ) {
  forn(i, n)
    if ((a[i] += b[i]) >= base)
      a[i] -= base, a[i + 1]++;
}
void sub( int *a, int *b ) {
  forn(i, n)
    if ((a[i] -= b[i]) < 0)
      a[i] += base, a[i + 1]--;
}
int cmp( int *a, int *b ) {
  forn(i, n)
```

```
    if (a[i] != b[i])
      return a[i] - b[i];
  return 0;
}
void mul( int *a, int k ) {
  forn(i, n)
    a[i] *= k;
  forn(i, n)
    if (a[i] >= base)
      a[i + 1] += a[i] / base, a[i] %= base;
}
```

3. **Числа со знаком.**
   Храним пару "знак и число". При сложении разных по знаку чисел сперва сравниваем их модули.
   Вещественные числа.
   Храним пару "сдвиг и число". При сложении двух чисел с разным сдвигом, сперва приводим их к одному сдвигу (дописываем в младшие разряды нули).

4. **Перевод из одной системы счисления в другую.**
   Мы умеем это делать за квадрат последовательным делением на короткое. Быстрее квадрата будет в ДЗ, на лекции рассказывать не нужно.

5. **mul, div, sqrt, gcd для чисел и для многочленов за квадрат.**

```
void mul(int *a, int *b, int *c) {
  int an = len(a), bn = len(b);
  fill(c, c + n, 0);
  for (int i = 0; i < an; i++)
    for (int i = 0; i < bn; i++)
      c[i + j] += a[i] * b[j];
  // и переносы сделать, если это число
}
void div(int *a, int *b, int *c) { // остаток -> a, частное -> c
  int an = len(a), bn = len(b);
  for (int i = an - 1; i >= bn - 1; i--) {
    while (a[i] >= b[bn - 1])
      c[i - (bn - 1)]++, sub(a, b, i); // вычтем со сдвигом
    a[i - 1] += a[i] * 10, a[i] = 0;
  }
num gcd( num a, num b ) {
  return isZero(b) ? a : gcd(b, a % b);
}
```

   Заметим, что `mul` работает за $\mathcal{O}(nm)$, а `div` работает за $\mathcal{O}((n-m)m)$. Из второго следует, что gcd работает за $\mathcal{O}(nm)$. Корень можно искать за куб бинпоиском. А можно последовательно вычисляя каждую следующую цифру за $\mathcal{O}(n)$, итого $\mathcal{O}(n^2)$.

6. **Реализация через вычитание и деление, умножение на два.**

```
num mul( num a, num b ) {
```

```
  if (isZero(b))
    return 0;
  num r = mul(a, b >> 1) << 1;
  if (b & 1)
    r += a;
  return r;
}
num div( num &a, num b ) {
  num c = div(a, b << 1) << 1;
  if (a >= b)
    c++, a -= b;
  return c;
}
num gcd( num a, num b ) {
  if (a % 2 == 0 && b % 2 == 0) // O(1)
    return gcd(a >> 1, b >> 1) << 1;
  if (isZero(b)) return a;
  if (isZero(a)) return b;
  while (b % 2 == 0) b >>= 1;
  while (a % 2 == 0) a >>= 1;
  if (a < b) swap(a, b):
  return gcd(a - b, b);
}
```

7. **Классный код для умножения чисел.**

```
inf = base * base; // base = 10^9
forn(i, an)
  forn(j, bn)
    if ((c[i + j] += a[i] * b[j]) >= inf)
      c[i + j] -= inf, c[i + j + 1] += base;
forn(i, an + bn)
  if (c[i] >= base)
    c[i + 1] += c[i] / base, c[i] %= base;
```

8. **Выбор системы счисления, чтобы всё работало за $\mathcal{O}(\frac{n^2}{k^2})$.**
Код умножения показан выше. Деление основано на том, чтобы вычитать не по одному, а сразу `a[i]/b[bn-1]`. В извлечении корня нужно за $\mathcal{O}(1)$ вычислять старшую цифру. $(C + base^i x)^2 = A$, где $C \approx \sqrt{A}$, x – цифра, которую предстоит найти. Пусть C состоит из хотя бы одной цифры, т.е. $C \cdot base^i x \geq base^{2i} x^2$, тогда пусть $x = (A - C^2)/(2C \cdot base^i) - 1$ (могли ошибиться максимум на 1, ничего страшного).

Используем FFT: div, sqrt

1. Разделяй и властвуй: div, sqrt за $\mathcal{O}(n \log^2 n)$.
2. Деление через вычисление $\frac{1}{P(x)}$ за $\mathcal{O}(n \log n)$.
3. sqrt за $\mathcal{O}(n \log n)$.