競技プログラミングのための FFT

多項式乗算の高速化ことはじめ

tsutaj (@_TTJR_)

Hokkaido University M1

June 24, 2018

本スライドの目標

多項式乗算

p 次多項式 $g(x) = \sum_{i=0}^{p} c_i x^i$ と、q 次多項式 $h(x) = \sum_{i=0}^{q} d_i x^i$ の乗算を以下のように定義

$$(g * h)(x) = g(x)h(x) = \sum_{k=0}^{p+q} \sum_{i=0}^{\min(k,p)} c_i d_{k-i} x^k \tag{1}$$

- 例: $g(x) = 2x + 3, h(x) = 3x^2 + 4x + 1$
  - $(g * h)(x) = (2x + 3)(3x^2 + 4x + 1) = 6x^3 + 17x^2 + 14x + 3$
- g も h も n 次式であるとしたとき、乗算にかかる計算量は $O(n^2)$ だが、FFT によって $O(n \log n)$ に高速化できる！
- 本スライドの目標: FFT を利用して多項式乗算を高速化すること
  - ほかの応用などには触れません

基本戦略

- p 次多項式 $g(x)$ と q 次多項式 $h(x)$ の積 → $p+q$ 次式
- m 次多項式 $f(x)$ に対して、少なくとも $m+1$ 個の点 $x_0, \ldots, x_m$ での値 $f(x_0), \ldots, f(x_m)$ が分かっていれば、これらを全て通るような多項式が一意に定まる
  - $m = 1$ (直線) の場合、2 点の値が分かっていれば一意に定まる
  - $m = 2$ (放物線) の場合、3 点の値が分かっていれば一意に定まる

基本戦略

1. 元の関数を求めるのに十分な数の点について、その関数値を求める
2. 設定した点を全て通るような関数を何らかの形で求める

- $p+q+1$ 個の点における関数値 が分かっていれば、$(g*h)(x)$ が分かりそう！
  - 「$(g*h)(x_i)$ を求める」=「$g(x_i)h(x_i)$ を求める」 なので計算は簡単
  - 計算において便利な点を定めて、楽に元の関数 $(g*h)(x)$ を得たい
  - 楽ができる点とは何か？

点の選び方

$N := (p+q+1) \leq N$ を満たす最小の 2 のべき乗 とし、N 個の点を用意することを考える

点の選び方

- 点 $x_0, \ldots, x_{N-1}$ として選ぶのは、1 の N 乗根！
- $\zeta_N = \exp(\frac{2\pi\sqrt{-1}}{N})$ とおくとき、$x_i = (\zeta_N)^i$ とすればよい

1 の N 乗根を点として選ぶと、何が嬉しいの？

- $(\zeta_N)^i = (\zeta_N)^j$ ならば、$i = j \mod N$ が成立
  - $(\zeta_N)^N = 1$ より、$(\zeta_N)^i = (\zeta_N)^{kN+i}$ $(k \in \mathbb{Z}^+)$ なので
- 直交性がある

$$\sum_{i=0}^{N-1} \left(\zeta_N^j\right)^i \left(\overline{\zeta_N^k}\right)^i = \sum_{i=0}^{N-1} \left(\zeta_N^j\right)^i \left(\zeta_N^{-k}\right)^i = \sum_{i=0}^{N-1} (\zeta_N)^{i(j-k)} \tag{2}$$

$$= \begin{cases} N & (j = k \mod N) \\ 0 & (\text{otherwise}) \end{cases} \tag{3}$$

係数の復元

説明のため $f(x) = (g * h)(x) = \sum_{j=0}^{N-1} h_j x^j$ とおく

- 先ほど選んだ点に対して、実際に $f\left(\zeta_N^0\right), \ldots, f\left(\zeta_N^i\right), \ldots, f\left(\zeta_N^{N-1}\right)$ を求めたとする
- ここから $f(x)$ の係数 $h_0, \ldots, h_{N-1}$ を復元するには？

離散フーリエ変換 (Discrete Fourier Transformation, DFT)

多項式 $f(x)$ に対して、選んだ各点での値を係数にもつ多項式 $\hat{f}(t)$ を作る
(下線部の f の置き換えに注意！)

$$\hat{f}(t) = \sum_{i=0}^{N-1} \underline{f\left(\zeta_N^i\right)} t^i = \sum_{i=0}^{N-1} \underline{\sum_{j=0}^{N-1}} h_j \left(\zeta_N^i\right)^j t^i \tag{4}$$

$$= \sum_{j=0}^{N-1} h_j \sum_{i=0}^{N-1} \left(\zeta_N^j t\right)^i \tag{5}$$

係数の復元

- DFT によって得られた $\hat{f}$ から係数を復元したい・・・
- $\hat{f}\left(\zeta_N^{-k}\right)$ を求めてみよう

$$\begin{aligned}\hat{f}\left(\zeta_N^{-k}\right) &= \sum_{j=0}^{N-1} h_j \sum_{i=0}^{N-1} \left(\zeta_N^j \zeta_N^{-k}\right)^i = \sum_{j=0}^{N-1} h_j \sum_{i=0}^{N-1} \left(\zeta_N\right)^{i(j-k)} \quad (6)\\ &= N \times h_k \quad (\zeta_N \text{ の直交性を利用}) \quad (7)\end{aligned}$$

- $\hat{f}\left(\zeta_N^{-k}\right)$ を求めることによって、なんと 元の関数 $f(x)$ の k 番目の係数 (を N 倍したもの) が求められる！ うれしい！
- $\hat{f}\left(\zeta_N^{0}\right), \ldots, \hat{f}\left(\zeta_N^{-(N-1)}\right)$ を求めてやればよさそう

離散フーリエ逆変換

離散フーリエ逆変換

$f(x)$ の DFT $\hat{f}(t)$ を利用して、元の関数 $f(x)$ は以下のように求められる

$$f(x) = \frac{1}{N} \sum_{i=0}^{N-1} \hat{f}\left(\zeta_N^{-i}\right) x^i \quad \left(\because \hat{f}\left(\zeta_N^{-i}\right) = N \times h_i\right) \tag{8}$$

- 離散フーリエ変換 (DFT)

$$\hat{f}(t) = \sum_{i=0}^{N-1} f\left(\zeta_N^{i}\right) t^i \tag{9}$$

  と比較すると、変数 x と t が入れ替わっていて、DFT で ζ_N^i であったところが ζ_N^{-i} へと変わっている
- 式 (8) を離散フーリエ逆変換 (inverse DFT) と呼ぶ

ここまでのまとめ

- 入力: p 次多項式 $g(x)$ と q 次多項式 $h(x)$
- 出力: 乗算して得られる多項式
  $f(x) = (g * h)(x) = g(x)h(x) = \sum_{j=0}^{N-1} h_j x^j$

離散フーリエ変換・逆変換を用いた解法

1. $N \leftarrow p + q + 1 \leq N$ を満たす最小の 2 のべき乗
2. 1 の N 乗根 $(\zeta_N)^0, \ldots, (\zeta_N)^{N-1}$ 全てに対し、$f\left(\zeta_N^i\right)$ を計算
3. 求めた関数値をもとに $\hat{f}(t) = \sum_{i=0}^{N-1} f\left(\zeta_N^i\right) t^i$ を作る (DFT)
4. $\hat{f}\left(\zeta_N^{-k}\right) = N \times h_k$ であることを利用して、元の多項式を復元 (inverse DFT)

- 残る課題は、 DFT と inverse DFT を高速に計算すること
- 高速に DFT を求めるアルゴリズム → 高速フーリエ変換 (Fast Fourier Transformation, FFT)

高速フーリエ変換

多項式 $f(x) = \sum_{i=0}^{N-1} h_i x^i$ (N は 2 のべき乗) を以下のように 2 つに分けてみよう

$$f_0(x) = \sum_{i=0}^{\frac{N}{2}-1} h_{2i} x^i = h_0 x^0 + h_2 x^1 + h_4 x^2 + \dots \tag{10}$$

$$f_1(x) = \sum_{i=0}^{\frac{N}{2}-1} h_{2i+1} x^i = h_1 x^0 + h_3 x^1 + h_5 x^2 + \dots \tag{11}$$

- 元の多項式は、 $f(x) = f_0(x^2) + x f_1(x^2)$ と表せる
- f_0, f_1 はそれぞれ $\frac{N}{2} - 1$ 次以下の多項式

高速フーリエ変換

- 先ほど $\hat{f}(t)$ を求めるには $f\left(\zeta_N^0\right), \ldots, f\left(\zeta_N^{N-1}\right)$ の値が必要だった
- 同様に考えると、 $\hat{f}_0(t), \hat{f}_1(t)$ を求めるには、
  $f(x) = f_0(x^2) + x f_1(x^2)$ より、以下が必要

$$f_0\left(\zeta_N^0\right), f_0\left(\zeta_N^2\right), \ldots, f_0\left(\zeta_N^{2(N-1)}\right) \tag{12}$$

$$f_1\left(\zeta_N^0\right), f_1\left(\zeta_N^2\right), \ldots, f_1\left(\zeta_N^{2(N-1)}\right) \tag{13}$$

高速フーリエ変換

$\zeta_N^2 = \exp\left(2 \times \frac{2\pi\sqrt{-1}}{N}\right) = \exp\left(\frac{2\pi\sqrt{-1}}{\frac{N}{2}}\right) = \zeta_{\frac{N}{2}}$ より、式 (12), (13) は以下のように変形可能

$$f_0\left(\zeta_{\frac{N}{2}}^0\right), f_0\left(\zeta_{\frac{N}{2}}^1\right), \ldots, f_0\left(\zeta_{\frac{N}{2}}^{N-1}\right) \tag{14}$$

$$f_1\left(\zeta_{\frac{N}{2}}^0\right), f_1\left(\zeta_{\frac{N}{2}}^1\right), \ldots, f_1\left(\zeta_{\frac{N}{2}}^{N-1}\right) \tag{15}$$

高速フーリエ変換

- ここで、$i = j \mod \frac{N}{2}$ ならば $\zeta_{\frac{N}{2}}^{i} = \zeta_{\frac{N}{2}}^{j}$
- 式 (14), (15) はそれぞれ前半と後半が同一視できる
- よって前半部分のみを求めればよく、具体的には以下を求めれば良い

$$f_0\left(\zeta_{\frac{N}{2}}^{0}\right), f_0\left(\zeta_{\frac{N}{2}}^{1}\right), \ldots, f_0\left(\zeta_{\frac{N}{2}}^{\frac{N}{2}-1}\right) \tag{16}$$

$$f_1\left(\zeta_{\frac{N}{2}}^{0}\right), f_1\left(\zeta_{\frac{N}{2}}^{1}\right), \ldots, f_1\left(\zeta_{\frac{N}{2}}^{\frac{N}{2}-1}\right) \tag{17}$$

高速フーリエ変換

目標: $\hat{f}(t)$ を求めるために必要な $f\left(\zeta_N^0\right), \ldots, f\left(\zeta_N^{N-1}\right)$ の値を求める

1. $f(x) = f_0(x^2) + x f_1(x^2)$ のように、f を 2 つの関数 f_0, f_1 に分ける
2. 2 つの多項式 f_0, f_1 に対して

$$f_0\left(\zeta_{\frac{N}{2}}^0\right), f_0\left(\zeta_{\frac{N}{2}}^1\right), \ldots, f_0\left(\zeta_{\frac{N}{2}}^{\frac{N}{2}-1}\right) \tag{18}$$

$$f_1\left(\zeta_{\frac{N}{2}}^0\right), f_1\left(\zeta_{\frac{N}{2}}^1\right), \ldots, f_1\left(\zeta_{\frac{N}{2}}^{\frac{N}{2}-1}\right) \tag{19}$$

   を計算することで、$f\left(\zeta_N^0\right), \ldots, f\left(\zeta_N^{N-1}\right)$ を求める

3. これを再帰的に実行 (サイズが半分になったものを 2 つ解くことの繰り返し)

高速フーリエ変換

- 前ページで説明した通りに再帰的に処理すると、これは計算量 $O(N \log N)$ である！
- inverse DFT に関しても、DFT で ζ_N^i であったところが ζ_N^{-i} へと変わっていて $\frac{1}{N}$ 倍されているだけなので、高速フーリエ変換と同様に高速に処理可能
- よって、全体で $O(N \log N)$ で解けた！！
- 実装例 Link

- END -