

算分回顾03-14

黄道吉-1600017857

元培学院

March 16, 2018

目录

- ① recap
- ② convolution
 - introduction
 - prerequisites
 - FFT and DFT⁻¹
- ③ convex hull
 - intro. and algo.
 - analysis
- ④ other problems

recap

- 最近点对
 - 预处理降低计算复杂度
 - 设计相应的数据结构
- 分治选择算法及其复杂度估计

vector operation

- 向量和 $a + b = (a_0 + b_0, \dots, a_{n-1} + b_{n-1})$
- 内积 $ab = (a_0b_0 + \dots + a_{n-1}b_{n-1})$
- 卷积 $ab = (c_0, \dots, c_{2n-2})$ 其中 $c_k = \sum_{i+j=k, i,j < n} a_ib_j$

•

$$\begin{pmatrix} a_0b_0 & a_0b_1 & \dots & a_0b_{n-2} & a_0b_{n-1} \\ a_1b_0 & a_1b_1 & \dots & a_1b_{n-2} & a_1b_{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ a_{n-1}b_0 & a_{n-1}b_1 & \dots & a_{n-1}b_{n-2} & a_{n-1}b_{n-1} \end{pmatrix}$$

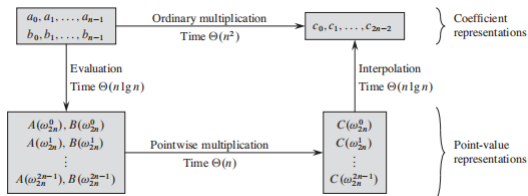
- 可以看作两个多项式相乘之后的各项系数

$$C(x) = A(x)B(x)$$

$$A(x) = \sum_{i=0}^{m-1} a_ix^i \quad B(x) = \sum_{i=0}^{n-1} b_ix^i$$

convolution computation

- 直接计算: $O(n^2)$
- 快速算法
 - 选择 x_1, \dots, x_{2n} , 求出 $A(x_j)$ 和 $B(x_j)$
 - 计算 $C(x_j) = A(x_j)B(x_j)$
 - 利用插值算法, 求出 $C(x)$
 - 下面只需要说明可以在 $\Theta(n \log n)$ 时间内对多项式求值, 反解出多项式



interpolation

- 可以用点值对表示多项式: 给定 $n-1$ 次多项式在 n 个点的取值, 可以唯一确定这个多项式, 反解出系数只需计算 $a = V(x_0, \dots, x_{n-1})^{-1}y$ 即可, 但这是 $O(n^3)$ 的算法

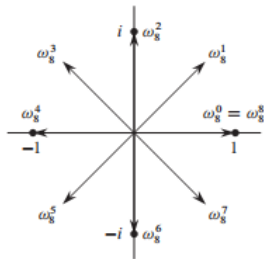
$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix}.$$

- 拉格朗日插值可以做到用 $\Theta(n^2)$ 的时间计算系数向量.

$$A(x) = \sum_{k=0}^{n-1} y_k \frac{\prod_{j \neq k} (x - x_j)}{\prod_{j \neq k} (x_k - x_j)}$$

complex roots of unity

- 1的单位根表示为 $\omega^n = 1$
- 在复平面上的表示



- 2n次方根的平方是n次方根, 即 $(\omega_{2n}^k)^2 = \omega_n^k$

DFT and FFT

- 假定一个多项式 $A(x) = \sum_{i=0}^{m-1} a_i x^i$ 以系数向量的形式给出 $a = (a_0, \dots, a_{n-1})$, 定义

$$y_k = A(\omega_n^k) = \sum_{j=0}^{n-1} a_j \omega_n^{kj}$$

则 $y = (y_0, \dots, y_{n-1}) = DFT_n(a)$ 称为系数向量的离散傅里叶变换

DFT and FFT

- 通过FFT可以在 $\Theta(n \log n)$ 时间内计算出DFT, 反解出系数向量
- 将多项式按照奇偶项分成两个 $n / 2$ 次多项式

$$A^{[0]}(x) = a_0 + \dots + a_{n-2}x^{n/2-1},$$

$$A^{[1]}(x) = a_1 + \dots + a_{n-1}x^{n/2-1}.$$

则有

$$A(x) = A^{[0]}(x^2) + xA^{[1]}(x^2)$$

DFT and FFT

- $A(x) = A^{[0]}(x^2) + xA^{[1]}(x^2)$
- 这样对 n 次多项式的求值转化为对两个 $n / 2$ 次多项式的求值
- 并且 $(\omega_n^0)^2, \dots, (\omega_n^{n-1})^2$ 正好是1的 $n / 2$ 次方根, 正好得到规模减小的原问题
- 合并子问题解的过程可以在 $O(n)$ 时间内完成, 也可以利用单位根的性质做优化.

FFT pseudo-code

Algorithm 1 Recursive-FFT(a)

```

1:  $n = a.length$ 
2: if  $n == 1$  then return  $a$ 
3: end if
4:  $\omega_n = e^{2\pi i/n}$        $\omega = 1$ 
5:  $a^{[0]} = (a_0, \dots, a_{n-2})$     $a^{[1]} = (a_1, \dots, a_{n-1})$ 
6:  $y^{[0]} = \text{Recursive-FFT}(a^{[0]})$     $y^{[1]} = \text{Recursive-FFT}(a^{[1]})$ 
7: for  $k = 0 \rightarrow n/2 - 1$  do
8:    $y_k = y_k^{[0]} + \omega y_k^{[1]}$ 
9:    $y_{k+n/2} = y_k^{[0]} - \omega y_k^{[1]}$ 
10:   $\omega = \omega \omega_n$ 
11: end for
12: return  $y$ 

```

DFT-1

- 上页中算法是 $\Theta(n \log n)$ 的算法
- 求DFT的过程等价于进行如下运算

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_n & \omega_n^2 & \omega_n^3 & \dots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \omega_n^6 & \dots & \omega_n^{2(n-1)} \\ 1 & \omega_n^3 & \omega_n^6 & \omega_n^9 & \dots & \omega_n^{3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \omega_n^{3(n-1)} & \dots & \omega_n^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

可验证, 上述矩阵的逆为 $A_{i,j}^{-1} = \omega_n^{-ij}/n$

- DFT的逆运算为 $a_j = \frac{1}{n} \sum_{k=0}^{n-1} y_k \omega_n^{-kj}$, 这也是多项式求值的形式, 也可以用 $\Theta(n \log n)$ 的时间完成.
- 算法导论中还介绍了一种常数更低的实现方式.

convex hull-intro. and algo.

- 问题 给定离散点集, 求最小的凸多边形包含所有点

convex hull-intro. and algo.

- 问题 给定离散点集, 求最小的凸多边形包含所有点
- 算法
 - 以 y_{max}, y_{min} 的连线分成两个点集
 - $deal(L)$
 - 将距离分界线最远点加入凸包, 连接两条新分界线
 - 考虑用新分界线划分成的两个新子问题
 - $deal(R)$

algorithm analysis

- 初始用直线分割 $O(n)$
- $deal(n)$ 的复杂度 $W(n) \leq W(n-1) + O(n)$
 - 找距离最远点 $O(n)$
 - 找三角形外面的点 $O(n)$
 - 新的子问题至多只有 $n-1$ 个点
- $T(n) = O(n) + W(n) = O(n^2)$

other problems

- 二分查找 $T(n) = T(n/2) + \Theta(1)$
- 棋盘覆盖

$$T(n) = \begin{cases} O(1) & k = 0 \\ 4T(k-1) + O(1) & k > 0 \end{cases}$$

- 幂的和 $T(n) = T(n/2) + \Theta(1)$