FFT,NTT,FWT

by FWT

前言

- ~~以下摘自洲神的日记：~~
  - 凡事总须研究，才会明白。曾经金鞋讲过，我也还记得，可是不甚清楚。我翻开资料一查，这资料没有署名，歪歪斜斜的每页上都写着“结论显然证明留作练习”几个字。我横竖睡不着，仔细看了半夜，才从字缝里看出字来，满本都写着两个字是“背板”！

FFT是什么，可以吃吗

- （序列下标均从0开始）你有一个长度为n的序列A，和一个长度为m的序列B，你需要算一个长度为$n+m-1$的序列C，其中$C_k = \sum_{i+j=k} A_i \cdot B_j$。FFT可以解决在$O(n\log n)$的时间内解决这样一个问题。

- 用生成函数来描述则是，给出一个$n-1$次多项式$A(x) = \sum_{i=0}^{n-1} a_i \cdot x^i$，和一个$m-1$次多项式$B(x) = \sum_{i=0}^{m-1} b_i \cdot x^i$，算出多项式$C(x) = A(x) \cdot B(x)$的各项的系数。

FFT - 点值表示法

- 任意一个次数不超过n的多项式，如果知道这个多项式$n+1$个不同的点值（即采样的位置不同），就可以唯一确定这个多项式。

- ~~证明找拉格朗日插值讲师~~
- 如果不考虑复杂度的话，可以用待定系数法+高斯消元把这个多项式求出来。

FFT - 点值表示法

- 点值表示有一个好处，就是如果我们现在的信息是$A\{(x_0, A(x_0)), (x_1, A(x_1)) \cdots (x_{n+\mathrm{m}-1}, A(x_{n+m-1}))\}$和$B\{(x_0, B(x_0)), (x_1, B(x_1)) \cdots (x_{n+m-1}, B(x_{n+m-1}))\}$，很快就可以算出$C\{(x_0, A(x_0) \cdot B(x_0)), (x_1, A(x_1) \cdot B(x_1)) \cdots (x_{n+m-1}, A(x_{n+m-1}) \cdot B(x_{n+m-1}))\}$。

- 所以能够快速系数→点值和点值→系数就可以了。
- 这就是我们要用FFT干的事情。

FFT - 复数相关的东西

- 复数：形如$a+bi$的数，其中$i=\sqrt{-1}$，a,b是实数。
- 复数的乘法定义为$(a+bi)\times(c+di)=ac-bd+(ad+bd)i$。
- 复平面：一个和平面直角坐标系长得差不多的东西，$a+bi$对应到复平面上就是点(a,b)。显然复平面上的点和复数是一一对应的。
- 模长：$r=\sqrt{a^2+b^2}$
- 幅角：θ等于实轴与从$(0,0)$出发经过(a,b)的射线的夹角。
- 所以复数又可以用二元组(r,θ)来表示，对应的数是$r\cdot(\cos\theta+i\cdot\sin\theta)$。

FFT - 复数相关的东西

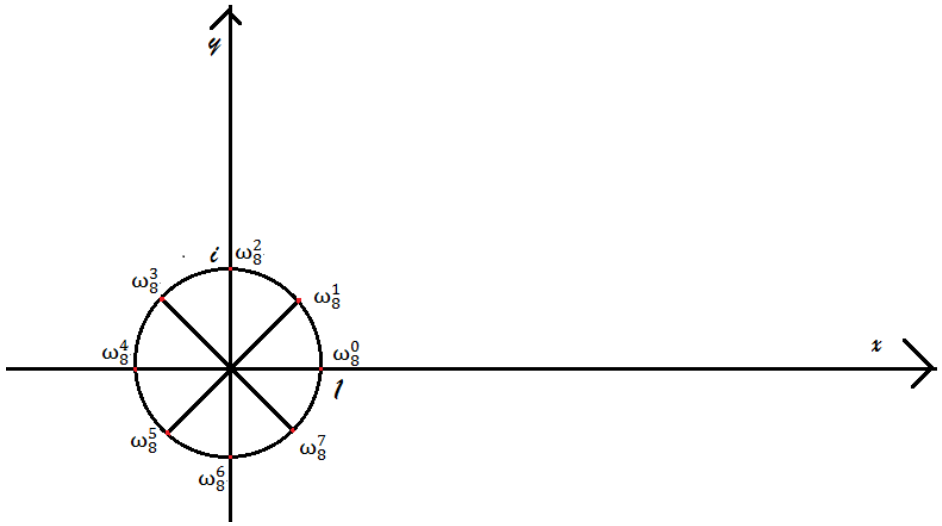
- 复数乘法满足：模长相乘，幅角相加。

设$z_1 = r_1(\cos\theta_1 + i \cdot \sin\theta_1), z_2 = r_2(\cos\theta_2 + i \cdot \sin\theta_2)$
则

$$z_1 \cdot z_2 = r_1 \cdot r_2[(\cos\theta_1 \cdot \cos\theta_2 - \sin\theta_1 \cdot \sin\theta_2) + i \cdot (\sin\theta_1 \cdot \cos\theta_2 + \sin\theta_2 \cdot \cos\theta_1)$$
$$= r_1 \cdot r_2[\cos(\theta_1 + \theta_2) + i \cdot \sin(\theta_1 + \theta_2)]$$

FFT - 复数相关的东西

- n次单位根：在复平面上，以原点为圆心，单位长度为半径作圆，所得的圆叫单位圆。考虑圆的 n 等分点（将 $(1,0)$ 作为第一个点），设幅角为正且最小的复数为ω_n，称为n次单位根。
- 根据复数的运算法则，其余的$n-1$个点可以分别表示为$\omega_n^2, \omega_n^3, \cdots \omega_n^n$，注意$\omega_n^n$与$\omega_n^0$是一样的。
- $\omega_n^0, \omega_n^1 \cdots \omega_n^{n-1}$便是方程$x^n = 1$的$n$个解。



FFT - 复数相关的东西

- 单位根有很多很好的性质。
- 周期性：$\omega_n^k = \omega_n^{k+n}$
- 对称性：$\omega_n^{k+\frac{n}{2}} = -\omega_n^k$
- $\omega_{mn}^{mk} = \omega_n^k$

FFT - 终于要开始变魔法啦！

- 假设现在要算点值的多项式是$A(x) = \sum_{i=0}^{n-1} a_i \cdot x^i$。
- 我们就令点值表达式里面的x取$x^n = 1$的n个解。
- 也就是说要求出一个序列$A' = \{A(\omega_n^0), A(\omega_n^1) \cdots A(\omega_n^{n-1})\}$。

- 下面只讨论n为2的整数次幂的情况。

FFT - 终于要开始变魔法啦！

- $A'_k = \sum_{i=0}^{n-1} a_i \cdot \omega_n^{ki} = \sum_{i\ is\ even}^{n-1} a_i \cdot \omega_{\frac{n}{2}}^{\frac{i}{2}k} + \omega_n^k \cdot \sum_{i\ is\ odd}^{n-1} a_i \cdot \omega_{\frac{n}{2}}^{\frac{i-1}{2}k}$

设

$$A_0(x) = a_0 + a_2 x^1 + a_4 x^2 + \cdots + a_{n-2} x^{\frac{n}{2}-1},$$

$$A_1(x) = a_1 + a_3 x^1 + a_5 x^2 + \cdots + a_{n-1} x^{\frac{n}{2}-1}$$

则$A(x) = A_0(x^2) + x \cdot A_1(x^2)$

- 对于$k < \frac{n}{2}$的情况，我们只需要知道A_0'和A_1'就可以算出来。

FFT - 终于要开始变魔法啦!

- 而对于$k \geq \frac{n}{2}$，发现:

$$A'_k = \sum_{i\ is\ even}^{n-1} a_i \cdot \omega_{\frac{n}{2}}^{\frac{i}{2}k} + \omega_n^k \cdot \sum_{i\ is\ odd}^{n-1} a_i \cdot \omega_{\frac{n}{2}}^{\frac{i-1}{2}k}$$

$$= \sum_{i\ is\ even}^{n-1} a_i \cdot \left(\omega_{\frac{n}{2}}^{k-\frac{n}{2}}\right)^{\frac{i}{2}} + \omega_n^k \cdot \sum_{i\ is\ odd}^{n-1} a_i \cdot \left(\omega_{\frac{n}{2}}^{k-\frac{n}{2}}\right)^{\frac{i-1}{2}}$$

所以知道A'_0, A_1'也就可以算出来了。
递归层数是$\log n$，所以复杂度是$O(n \log n)$。

FFT - 变回去

- （利用数列反演或者构造逆矩阵的思想）
- 可以证明:
- $a_k = \frac{1}{n}\sum_{i=0}^{n-1} A'_i \cdot \omega_n^{-ki}$
- 这个式子和前面的变换的形式是一样的，把ω_n^k替换成ω_n^{-k}，最后再除以一个n就可以了。

FFT - 变回去

$$a_k = \frac{1}{n}\sum_{i=0}^{n-1} A'_i \cdot \omega_n^{-ki} = \frac{1}{n}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_j\, \omega_n^{i(j-k)}$$

而$\sum_{i=0}^{n-1} \omega_n^{im}$满足当$m = 0$时为$n$，$m \neq 0$的时候为$0$。上式得证。

- 可以想一下这个玩意怎么证明。

FFT - 变回去

- 证：若$m = 0$则$\sum_{i=0}^{n-1} \omega_n^{im} = \sum_{i=0}^{n-1} 1^i = n$。
- 若$m \neq 0$则可以用等比数列求和公式，得到

$$\sum_{i=0}^{n-1} \omega_n^{im} = \frac{\omega_n^{nm}-1}{\omega_n^m} = 0$$

- 我相信，至此大家已经完全明白了如何用FFT计算多项式乘法辣(〃'▽'〃)

FFT - 优化

- FFT本身常数就比较大（因为是复数运算，还涉及到三角函数），如果写递归的话~~基本上就没救子~~。。。可以预先处理出每个数在整个递归分治的过程中最终会被放到的位置（即每一层将偶数下标的放左边，奇数下标的放右边，最后每个数会在的位置），然后直接迭代求解。
- 还有就是因为$\omega_n^k = -\omega_n^{k+\frac{n}{2}}$，所以可以少算一次复数的乘法。
- 遍历数组的时候可以用指针，有没有用取决于评测机。
- 注意long double精度比double好，但是会比double慢。

啊我饿了我要吃NTT

- 如果要求多项式的系数对某个数取模怎么办？单位根是没有办法在模意义下算的。
- 一种思路是算出精确值之后再取模。可以参考后面的MTT部分。

- 但是伟大的~~卡兹sama~~敏磊开始思考了：能不能直接在模意义下计算呢？(-_-)ゞ

- 于是NTT出世了。

NTT - 原根

- 对于一个质数p，定义g为它的原根，当且仅当$g^0, g^1, \cdots g^{p-2}$在模p意义下互不相同。（其实原根对非素数也是有定义的，并且有一个快速判定原根的算法，大家下来可以自己去了解一下）
- 注意到，$g^0, g^1, \cdots g^{p-2}$这些数的$p-1$次方模p都是1（费马小定理）。

- 于是我们可以用$g^{\frac{p-1}{n}}$来做ω_n。因为：1.$\left(g^{\frac{p-1}{n}}\right)^n \equiv 1$，这意味着它满足周期性和对称性；2.它的幂次互不相同。
- 然后我们就讲完NTT辣。

NTT - 模数

- 因为n必须是2^k，所以能够NTT的模数必须是形式为$t \cdot 2^p + 1$且存在原根的素数。
- 常用的：
  - 998244353，原根是3。
  - 1004535809，原根是3。

NTT - 优化

- 预处理出过程中可能用到的原根次幂，可以少算若干次快速幂。
- 然后就是关于模意义下的乘法、加法可能用到的系列优化。注意有的优化可能会让程序在某些机器上更慢。

FWT很难吃的(╥﹏╥)

- FWT有三种，分别对应了下面的三种卷积。

$$C_k = \sum_{i \text{ or } j=k} A_i \times B_j$$

$$C_k = \sum_{i \text{ and } j=k} A_i \times B_j$$

$$C_k = \sum_{i \text{ xor } j=k} A_i \times B_j$$

或卷积

- 令$A'_i = \sum_j A_j[i \mid j = i]$，则有$A'_i \times B'_i = C'_i$，这是因为$[i \mid k = k$并且$j \mid k = k] \Leftrightarrow [i \mid j \mid k = k]$。
- 剩下的问题就是如何快速完成$A \to A', A' \to A$。

或卷积

- 分别考虑A下标的最高位为0的和最高位为1的，设它们为A_0, A_1。
- 在不考虑最高位的情况下，对A_0, A_1做变换得到A_0', A_1'。
- 假设最高位是2^t，则最高位为0的数$A_i' = A'_{0_i}$，最高位为1的数$A_i' = A'_{0_{i-2^t}} + A'_{1_i}$。

- 逆变换就是对于最高位为0的数$A_i = A_{0_i}$，最高位为1的数$A_i = A_{1_i} - A_{0_i}$。

与卷积

- （其实可以看作下标取反之后的或卷积

- 定义$A'_i = \sum_j A_j[j\&i = i]$，则有$C'_i = A'_i \times B'_i$。

- 对于最高位为0的，$A'_i = {A_0}'_i + {A_1}'_{i+2^t}, A_i = A_{0i} - A_{1_{i+2^t}}$
- 对于最高位为1的，$A'_i = {A_1}'_i, A_i = A_{1_i}$

异或卷积

- ~~前面两个其实基本上不考~~

- 给出卷积的定义：定义$count(x)$表示x的二进制表示下的1的个数的奇偶性（1表示奇数，0表示偶数），则$A_i{}' = \sum_j A_j(-1)^{count(i\&j)}$。

- 可证明$C_i' = A_i' \times B_i'$。

异或卷积

$$C'_k = \sum_i A_i(-1)^{count(i\&k)} \sum_j B_j(-1)^{count(j\&k)}$$

$$= \sum_i \sum_j A_i B_j(-1)^{count(i\&k)\ \mathrm{xor}\ count(j\&k)}$$

$$= \sum_i \sum_j A_i B_j(-1)^{count((i\ \mathrm{xor}\ j)\&k)}$$

异或卷积

- 对于最高位为0的，$A_i{}' = A'_{0\,i} + A_{1}{}'_{i+2^t}$。
- 对于最高位为1的，$A_i{}' = A'_{0\,i-2^t} - A_{1}{}'_{i}$。

- 逆变换：
- 对于最高位为0的，$A_i = \frac{1}{2}(A_{0\ i} + A_{1\,i+2^t})$。
- 对于最高位为1的，$A_i = \frac{1}{2}(A_{0\ i-2^t} - A_{1\,i}\ )$。

一个很本质的问题

- 观察发现，前面的每一种变换，本质上都是构造了一个变换系数$trans(p,i)$，令$A'_i=\sum_j trans(i,j)\cdot A_j$，而我们要让$trans(p,i)\times trans(p,j)=trans(p,i\ \mathrm{opt}\ j)$成立。

另外一种卷积

- 形如$C_k = \sum_{j-i=k} A_i \cdot B_j$。
- 令$A'_i = A_{n-1-i}$，算出

子集卷积

- 定义为$C_S = \sum_X \sum_Y A_X \cdot B_Y [X \cup Y = S, X \cap Y = \emptyset]$。直接做的复杂度是$O(3^n)$。
- 注意到：

$$X \cup Y = S, X \cap Y = \emptyset \Leftrightarrow X \cup Y = S, |X| + |Y| = |S|$$

- 这样就可以先枚举$|X|, |Y|$，再做或卷积，时间复杂度$O(n^2 2^n)$。实现的时候注意是直接存点值，因为对点值加减等价于多项式加减。

MTT

- 如果数字特别大，虽然long double可以存下最后的答案，但是FFT的过程中会爆精度，怎么办？
- 将每一个数表示成$aM+b$的形式，其中$a,b\in[0,M)$，而M取$\sqrt{值域}$。

$$\begin{aligned}&\left(A_1(x)M+A_0(x)\right)\cdot\left(B_1(x)M+B_0(x)\right)\\&=A_1(x)\cdot B_1(x)\cdot M^2+\left(A_1(x)\cdot B_0(x)+A_0(x)\cdot B_1(x)\right)\cdot M+A_0(x)\\&\cdot B_0(x)\end{aligned}$$

- 分别对每个部分用FFT算就可以了。

循环卷积

- 形如$C_k = \sum_{i+j \equiv k \bmod n} A_i \cdot B_j$。
- 由于$\omega_n^{i+j} = \omega_n^{i+j \bmod n}$，所以做卷积的时候不要在高位填零，直接以$n$作为多项式的次数，就可以满足$C_i' = A_i' \times B_i'$。

- 顺便提一句，如果n不是2的整数次幂也是可以做的。
- 考虑到$ij = \binom{i+j}{2} - \binom{i}{2} - \binom{j}{2}$，则

$$A_i' = \sum_j A_j \omega_n^{ij} = \omega_n^{-\binom{i}{2}} \sum_j A_j \cdot \omega_n^{-\binom{j}{2}} \cdot \omega_n^{\binom{i+j}{2}}$$

这是个卷积的形式可以直接计算。

讲完了

- 如果想看更加详细的资料可以去看我写的学习笔记。

- 模板题：
  - 51nod1773 A国的贸易（异或卷积）
  - BZOJ4589 Hard Nim（异或卷积）
  - Card Game CSU - 1911（或卷积）
  - WC2018 州区划分（子集卷积）

LJZ AK WC!

- 蟹蟹大家ヾ(❀^ω^)ﾉﾞ