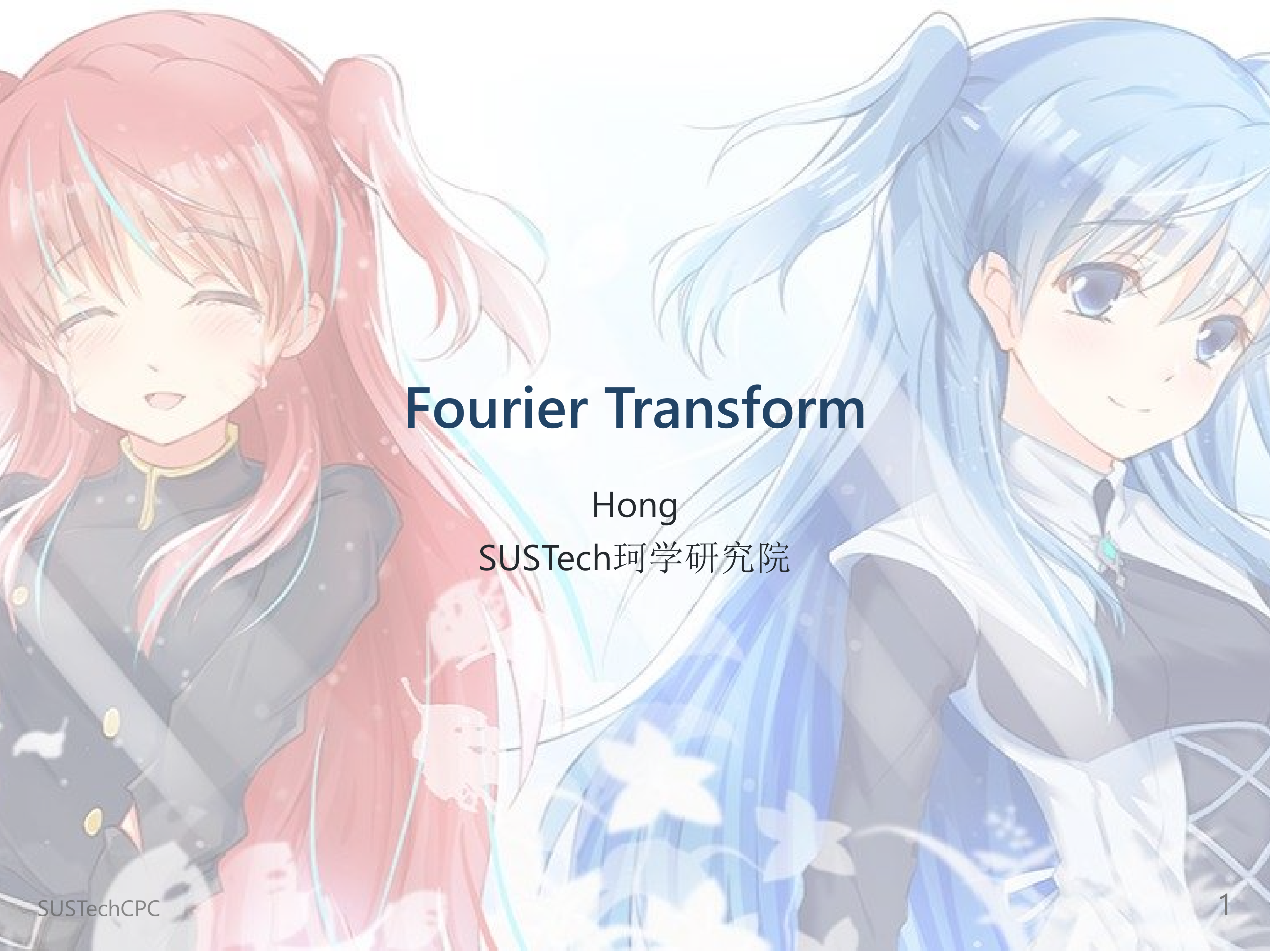

Fourier Transform

Hong
SUSTech珂学研究院

多项式乘法

给出多项式$A(x) = \sum a_n x^n$和$B(x) = \sum b_n x^n$，求$A(x) \cdot B(x)$

系数表示法和点值表示法

对于$N-1$次多项式$f(x)$，可以有2种不同的表示方法：

系数表示法：

$$f(x)=\sum_{0\le n<N} c_n x^n$$

点值表示法：

$(x_1,x_2,...,x_n),(y_1,y_2,...,y_n)$满足$f(x_i)=y_i$，即$(x_i,y_i)$是曲线$y=f(x)$上的点

显然，系数表示法和点值表示法可以相互转换。

系数表示法→点值表示法

$$y_i = \sum_{0\le n<N} c_i x^n$$

点值表示法→系数表示法

$$f(x) = \sum_{0\le n<N} y_i \frac{\Pi_{j\neq i}(x - x_j)}{\Pi_{j\neq i}(x_i - x_j)}$$

对于多项式$A(x)$和$B(x)$，假设$degA + degB < N$。

如果有A和B在$\{x_0, x_1, ..., x_{N-1}\}$处的点值表示，则$(A \cdot B)$的点值表示可以通过

$$(A \cdot B)(x_i) = A(x_i) \cdot B(x_i)$$

在$O(N)$时间内得到。

还原$(A \cdot B)$为系数表示就实现了多项式乘法。

变换复杂度为$O(N^2)$。

离散傅里叶变换(DFT)

考虑在$1, \omega, \omega^2, \ldots, \omega^{N-1}$的点值表示($\omega$为$N$次单位复根)，即

$$y_k = \sum_{i=0}^{N-1} c_i (\omega^k)^i$$

假设$N = 2^k$，$(y_0, y_1, \ldots, y_{N-1})$可以快速求出。

快速傅里叶变换(FFT)

$$f(x)=\sum_{i=0}^{N/2-1}a_{2i}(x^2)^i+x\sum_{i=0}^{N/2-1}a_{2i+1}(x^2)^i$$

当x取遍所有N次单位复根时，x^2取遍所有$N/2$次单位复根。所以只需要计算多项式

$$f_0(x)=\sum_{i=0}^{N/2-1}a_{2i}x^i$$

和

$$f_1(x)=\sum_{i=0}^{N/2-1}a_{2i+1}x^i$$

的DFT。

逆离散傅里叶变换(IDFT)

$$y_k = \sum_{i=0}^{N-1} c_i (\omega_k)^i$$

$$\frac{1}{N}\sum_{i=0}^{N-1} y_i(\omega^{-k})^i = \frac{1}{N}\sum_{i=0}^{N-1}(\sum_{i=0}^{N-1} c_j(\omega^i)^j)(\omega^{-k})^i$$

$$= \frac{1}{N}\sum_{j=0}^{N-1} c_j(\sum_{i=0}^{N-1}(\omega^{j-k})^i) = c_k$$

当$k \neq 0$时，

$$1 + \omega^k + (\omega^k)^2 + \ldots + (\omega^k)^{N-1} = \frac{1-(\omega^k)^N}{1-\omega^k} = 0$$

卷积下的FFT

- 卷积：给定有限长度的序列a_i, b_i，求序列c使得 $c_i = \sum_{k=0}^{i} a_k b_{i-k}$，显然$c$的长度为$a, b$的长度之和减一。
- $c_r = \sum_{p,q}[(p+q)\ mod\ n = r]a_p b_q$
- $\frac{1}{n}\sum_{k=0}^{n-1}\omega^{vk} = [v\ mod\ n = 0]$
- $[(p+q)\ mod\ n = r] = [(p+q-r)\ mod\ n = 0]$ $= \frac{1}{n}\sum_{k=0}^{n-1}\omega^{(p+q-r)k} = \frac{1}{n}\sum_{k=0}^{n-1}\omega^{-rk}\omega^{pk}\omega^{qk}$
- 所以有$c_r = \sum_{p,q}[(p+q)\ mod\ n = r]a_p b_q$ $= \frac{1}{n}\sum_{k=0}^{n-1}\omega^{-rk}\sum_p \omega^{pk}a_p \sum_q \omega^{qk}b_q$
- 使用前面的FFT求解后两个求和式，再求解第一个求和式即可。

蝴蝶操作与位逆序置换

对奇偶项分开递归处理的操作称为蝴蝶操作。

考虑每个函数的系数：

$$(a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7)$$

$$(a_0, a_2, a_4, a_6)(a_1, a_3, a_5, a_7)$$

$$(a_0, a_4)(a_2, a_6)(a_1, a_5)(a_3, a_7)$$

$$(a_0)(a_4)(a_2)(a_6)(a_1)(a_5)(a_3)(a_7)$$

观察$(0, 4, 2, 6, 1, 5, 3, 7)$的二进制表示为$(000, 100, 010, 110, 001, 101, 011, 111)$，恰好是把$(000, 001, 010, 011, 100, 101, 110, 111)$每一位二进制反转，称为位逆序置换。这样可以实现非递归的FFT，常数更小。

快速数论变换(NTT)

$\{1, \omega, \omega^2, \omega^3, ...\}$是$2^k$阶的循环群。

对于质数$P = 2^K \cdot Q + 1$的原根g，$\{1, g, g^2, g^3, ...\}$是$2^K \cdot Q$阶的循环群。

即$\{1, g^Q, g^{2Q}, g^{3Q}, ...\}$是$2^K$阶循环群。用$g$替代$\omega$即可。

快速数论变换仅适用于数论函数，即所有参数均为自然数的情况。
但快速数论变换比快速傅里叶变换常数更小。

Triple Sums

N个整数$A_1, A_2, ..., A_N$，对于所有的S，求满足:

- $A_i + A_j + A_k = S$
- $i < j < k$

的(i, j, k)数量。

$(N \le 40000, A_i \le 20000)$

考虑(i,j,k)可以相等的情况，构造多项式

$$A(x)=\sum_{1\le i\le N} x^{A_i}$$

则$A^3(x)$中x^S项的系数就是所求结果。

考虑使用容斥原理消去相等的影响

$$(\sum x)^3 = \sum x^3 + 3\sum x^2 y + 6\sum xyz$$

$$(\sum x^2)(\sum x) = \sum x^3 + \sum x^2 y$$

即

$$\sum xyz = \frac{(\sum x)^3 - 3(\sum x^2)(\sum x) + 2(\sum x^3)}{6}$$

NTT运算即可

(SPOJ TSUM)

Pattern matching

长度为m的模式串A和长度为n文本串B包含小写字母和通配符'*','*'可以匹配任意一个字符.

求A在B中所有的出现位置。

对于两个长度相等的字符串A和B,定义距离函数:

$$dis(A,B)=\sum_{i=0}^{N-1}(A[i]-B[i])^2$$

若$dis(A,B)==0$，则A和B完全匹配。

考虑通配符，设通配符值为0，可将距离函数修改为:

$$dis(A,B)=\sum_{i=0}^{N-1}(A[i]-B[i])^2\cdot A[i]\cdot B[i]$$

若$dis(A,B)==0$，则A和B完全匹配。

考虑文本串的末尾位置i，设$f[i] = dis(A, B[i-m+1, i])$。

则若以i结尾的子串和A匹配，有：

$$f[i] = \sum_{j=0}^{m-1} (A[j] - B[i-m+1+j])^2 A[j] B[i-m+1+j] = 0$$

将A串翻转，并在后面不断补0直至和B串等长，那么有：

$$f[i]=\sum_{j=0}^{i}(A[j]-B[i-j])^2A[j]B[i-j]$$

$$=\sum_{j=0}^{i}(A[j]^2-2A[j]B[i-j]+B[i-j]^2)A[j]B[i-j]$$

$$=\sum_{j=0}^{i}A[j]^3B[i-j]-2\sum_{j=0}^{i}A[j]^2B[i-j]^2+\sum_{j=0}^{i}A[j]B[i-j]^3$$

显然可以分三段做FFT求出所有的$f[i]$，时间复杂度为$O(nlogn)$。

存在一种更简单的构造方法。

令

$$dist(A,B)=\sum_{i=0}^{N-1}A[i]\cdot inv(B[i])$$

若$dist(A,B)\le N$，则匹配，否则不匹配。只需要做一次FFT或NTT即可。

(BZOJ4259)