

# FWT

## 简介

$\mathrm{FWT}$  (快速沃尔什变换)，是用来处理将按位运算作为操作符的多项式卷积，以及其衍生出来的一些问题。

在接触之前，你必须已经基本掌握  $\mathrm{FFT}$  (快速傅里叶变换)，两者的原理内核基本一致。

## 算法内容

### 大概思路

$\mathrm{FWT}$  的基础功能，是用来处理下面的求和式子 
$$\begin{array}{l} C_k = \sum_{i+j=k} A_i * B_j \\ C_k = \sum_{i \& j=k} A_i * B_j \\ C_k = \sum_{i \wedge j=k} A_i * B_j \end{array}$$
 这样的式子虽然看似卷积，但没有办法用  $\mathrm{FFT}$  来做。

我们先想想  $\mathrm{FFT}$  的工作：先对于一个多项式求出他在若干个单位根的点值表示法，再将多项式乘起来，最后再复原。那么我们实际上也可以有类似的思路：能否先将多项式求出另外一个多项式  $\mathrm{FWT}(A)$  再将对应的位置乘起来，最后再复原？

也就是  $\mathrm{FWT}(C) = \mathrm{FWT}(A) * \mathrm{FWT}(B)$  继续讨论前，我们先定义以下的东西 
$$\begin{aligned} A+B &= \left( A_0+B_0, A_1+B_1, \dots \right) \\ A-B &= \left( A_0-B_0, A_1-B_1, \dots \right) \\ A \oplus B &= \left( \sum_{i \oplus j=0} A_i * B_j, \sum_{i \oplus j=1} A_i * B_j, \dots \right) \end{aligned}$$

### 或 (or) 卷积

或卷积写成向量的形式  $A \oplus B = \left( \sum_{i+j=0} A_i * B_j, \sum_{i+j=1} A_i * B_j, \dots \right)$  很显然，这个式子满足交换律  $A \oplus B = B \oplus A$  事实上也满足分配律  $(A+B) \oplus C = \left( \sum_{i+j=0} (A_i+B_i) * C_j, \dots \right)$  很显然可以把括号拆开分成两个  $\sum$  变成  $A \oplus C + B \oplus C$

定义1：我们这样定义  $\mathrm{FWT}$  映射  $\mathrm{FWT}(A)_i = \sum_{j|i} A_j$  这个式子可以靠感性理解其成立，严谨证明如下，下面  $j|i$  代表  $j$  是  $i$  的因子 
$$\begin{aligned} \mathrm{FWT}(C)_i &= \sum_{j|i} C_j = \sum_{j|i} \sum_{x+y=j} A_x \times B_y = \sum_{(x|y)|i} A_x \times B_y \\ &= \sum_{x|i} A_x \times \sum_{y|i} B_y = \mathrm{FWT}(A)_i \times \mathrm{FWT}(B)_i \end{aligned}$$
 定义2：等价定义为，对于一个多项式  $A$  最高次项为  $2^n$  我们把它分成两部分  $A_0, A_1$  分别表示前面  $2^{n-1}$  次项和后面  $2^{n-1}$  次项，也就是下标最高位为  $0$  与  $1$  两个部分 
$$\mathrm{FWT}(A) = \left( \left( \mathrm{FWT}(A_0) \right), \mathrm{FWT}(A_0+A_1) \right) \quad n > 0$$
 
$$\mathrm{FWT}(A) = \left( \mathrm{FWT}(A_0) \right) \quad n = 0$$
 其中的括号与逗号表示多项式拼接。

两者的等价性证明从简，其实是容易看出，如果定义2前式右边是  $\mathrm{FWT}(A_0) + \mathrm{FWT}(A_1)$  那么定义1是满足定义2的，而定义2

