Notes on Binius (Part II): Subspace Polynomial

- Yu Guo yu.guo@secbit.io
- Jade Xie jade@secbit.io

FRI-Binus 论文 [DP24] 中讨论了基于 Subspace Polynomial 的 Additive FFT 算法，并给出了用奇偶项分解的视角来理解 [LCH14] 中的基于 Novel Polynomial Basis 的 Additive FFT 算法。本文直接介绍子空间多项式 Subspace Polynomial，然后据此介绍奇偶项分解视角的 Additive FFT 算法。本文省去了 Normalized Subspace Polynomial 的定义，方便读者理解。Normalization 只是影响 FFT 算法的性能，和本文介绍的简化版算法并没有本质的区别。

由于 Additive FFT 依赖的代数结构和素数域上的 Multiplicative FFT 非常相似，因此关于 Multiplicative FFT 的知识将有助于理解本文的内容。

线性子空间多项式 Subspace Polynomial

我们继续探索基于 $\mathbb{F}_2$ 上的 Extension Field，$\mathbb{F}_{2^m}$ 。不管是用何种方式构造的 $\mathbb{F}_{2^m}$，所有的元素构成了一个向量空间，记为 V_m ，并且存在一组 Basis $(\beta_0, \beta_1, \ldots, \beta_{m-1})$ ，张成这个向量空间，记为 $V_m = \mathsf{Span}(\beta_0, \beta_1, \ldots, \beta_{m-1})$，或者用符号 $\langle \cdots \rangle$ 表示：

$$V_m = \langle \beta_0, \beta_1, \ldots, \beta_{m-1} \rangle \tag{1}$$

这样任意一个元素 $\theta \in \mathbb{F}_{2^m}$ ，可以写为 Basis 分量的线性组合：

$$\theta = c_0 \cdot \beta_0 + c_1 \cdot \beta_1 + \ldots + c_{m-1} \cdot \beta_{m-1}, \text{ where } c_i \in \mathbb{F}_2 \tag{2}$$

同时 V_m 还是一个加法群，单位元为 $V_0 = \{0\}$ ，如果 V_k 是 V_m 的一个线性子空间，那么 V_k 也是 V_m 的一个加法子群。对于 V_k ，我们可以用一个多项式来编码其中的所有元素，即该多项式的根集合正好对应 V_k 的所有元素的集合，我们把多项式记为 $s_k(X)$ 。这个多项式也被称为「Subspace Polynomial」子空间多项式：

$$s_k(X) = \prod_{i=0}^{2^k-1} (X - \theta_i), \text{ where } \theta_i \in V_k \tag{3}$$

多项式 $s_k(X)$ 也可以看成是 V_k 这个 Domain 上的 Vanishing Polynomial，因为对于任意的 $\theta \in V_k$ ，都满足：

$$s_k(\theta) = 0 \tag{4}$$

Linearized Polynomial

上面介绍的 Subspace 多项式是一种所谓的 Linearized Polynomial，因为它的定义满足下面的形式：

$$L(X) = \sum_{i=0}^{n-1} c_i \cdot X^{q^i}, \quad c_i \in \mathbb{F}_q \tag{5}$$

多项式 $L(X)$ 之所以被称为 Linearized Polynomial，因为每一个 $L(X)$ 都对应到 $\mathbb{F}_q$ 的扩张域 K 上的一个线性算子（Linear Operator）。假如 $L(X)$ 的所有根都在扩张域 $K = \mathbb{F}_{q^s}$ 中，那么对于所有的 $\theta \in K$，都有 $L(\theta) \in K$。而且，如果 $\theta \neq \theta'$ ，那么 $L(\theta) \neq L(\theta')$。每一个 $L(X)$ 都可以被视为一个矩阵 $B \in \mathbb{F}_q^{s \times s}$ ，完成 向量空间 $\mathbb{F}_q^s$ 上的线性变换，使得：

$$(c_0, c_1, \ldots, c_{s-1})B = (d_0, d_1, \ldots, d_{s-1}) \tag{6}$$

对于 Subspace Polynomial 而言，每一个 $s_k(X)$ 都是一个 Linearized Polynomial，反过来，任何一个 Linearized polynomial $L(X) \in \mathbb{F}_{q^m}[X]$，它的所有根都构成某个线性子空间 $V_n \subset V_m$。详细的证明过程请参考 [LN97]。

线性性质

由于 $s_k(X)$ 的每一项都是 $a_i \cdot X^{2^i}$ 的形式，因此它具有加法的同态性：

$$
\begin{aligned}
s_k(x+y) &= s_k(x) + s_k(y), && \forall x, y \in \mathbb{F}_{2^m} \\
s_k(c \cdot x) &= c \cdot s_k(x), && \forall x \in \mathbb{F}_{2^m}, \forall c \in \mathbb{F}_2
\end{aligned}
$$

我们来尝试简单证明第一个等式，根据有限域理论的一个常见定理（Freshman's dream）：

$$
(x+y)^2 = x^2 + 2xy + y^2 = x^2 + y^2, \quad \text{where } x, y \in \mathbb{F}_{2^m} \tag{7}
$$

显然，$2xy = 0$，因为在二进制域中，$2 = 0$。所以下面的等式也同理成立：

$$
(x+y)^{2^i} = x^{2^i} + y^{2^i} \tag{8}
$$

接下来验证下 $s_k(X)$ 的加法同态性：

$$
s_k(x+y) = \sum_{i=0}^{k} a_i \cdot (x+y)^{2^i} = \sum_{i=0}^{k} a_i \cdot \left(x^{2^i} + y^{2^i}\right) = s_k(x) + s_k(y) \tag{9}
$$

子空间多项式的递推式

对于子空间 V_k，它可以被拆分为两个不相交的集合：

$$
V_k = V_{k-1} \cup (\beta_{k-1} + V_{k-1}) \tag{10}
$$

这里 $V_k = \langle \beta_0, \beta_1, \ldots, \beta_{k-1} \rangle$, $V_{k-1} = \langle \beta_0, \beta_1, \ldots, \beta_{k-2} \rangle$ ， 那么 V_k, V_{k-1}, $\beta_{k-1} + V_{k-1}$ 所对应的子空间多项式满足下面的关系：

$$
s_k(X) = s_{k-1}(X) \cdot s_{k-1}(X + \beta_{k-1}) \tag{11}
$$

举个简单例子，假设 $k = 3$，$V_3 = \langle \beta_0, \beta_1, \beta_2 \rangle$ 由两部分构成，一部分是 $V_2 = \langle \beta_0, \beta_1 \rangle$，另一部分是 V_2 中的每一个元素加上 β_2。因此，V_3 的元素个数为 $2^2 + 2^2 = 8$，下面列出 V_3 的全部元素：

$$
V_3 = \{0, \beta_0, \beta_1, \beta_0 + \beta_1\} \cup \{\beta_2, \beta_0 + \beta_2, \beta_1 + \beta_2, (\beta_0 + \beta_1) + \beta_2\} \tag{12}
$$

我们容易验证： $s_3(X) = s_2(X) \cdot s_2(X + \beta_{k-1})$ 。当然 $s_2(X)$ 也可以拆成关于 $s_1(X)$ 和 $s_1(X + \beta_1)$ 的乘积，我们不妨试着拆解到底：

$$
\begin{aligned}
s_3(X) &= s_2(X) \cdot s_2(X + \beta_2) \\
&= s_2(X)^2 + \beta_2 \cdot s_2(X) \\
&= s_1(X) \cdot s_1(X + \beta_1) \cdot s_1(X) \cdot s_1(X + \beta_1) + \beta_2 \cdot s_1(X) \cdot s_1(X + \beta_1) \\
&= (s_1(X)^2 + \beta_1 \cdot s_1(X))^2 + \beta_2 \cdot s_1(X)^2 + \beta_1\beta_2 \cdot s_1(X) \\
&= s_1(X)^4 + \beta_1^2 \cdot s_1(X)^2 + \beta_2 \cdot s_1(X)^2 + \beta_1\beta_2 \cdot s_1(X) \\
&= s_1(X)^4 + (\beta_1^2 + \beta_2) \cdot s_1(X)^2 + \beta_1\beta_2 \cdot s_1(X) \\
&= (X \cdot (X + \beta_0))^4 + (\beta_1^2 + \beta_2) \cdot (X \cdot (X + \beta_0))^2 + \beta_1\beta_2 \cdot (X \cdot (X + \beta_0)) \\
&= (X^2 + \beta_0 \cdot X)^4 + (\beta_1^2 + \beta_2) \cdot (X^2 + \beta_0 \cdot X)^2 + \beta_1\beta_2 \cdot (X^2 + \beta_0 \cdot X) \\
&= X^8 + \beta_0^4 X^4 + (\beta_1^2 + \beta_2) X^4 + \beta_0^2(\beta_1^2 + \beta_2) X^2 + \beta_1\beta_2 X^2 + \beta_0\beta_1\beta_2 X
\end{aligned} \tag{13}
$$

最后 $s_3(X)$ 的展开式满足 $\sum_{i=0}^{k} a_i \cdot X^{2^i}$ 这样的模式，也符合了我们上面的结论。

子空间上的同态映射

因为 Subspace Polynomial 实际上是一种 Vanishing Polynomial，并且它还具有加法同态，所以我们可以利用 Subspace Polynomial 来定义子空间之间的同态映射。

例如对于 $V_3 = \langle \beta_0, \beta_1, \beta_2 \rangle$ ，我们定义 V_3 的子空间 $V_1 = \{0, \beta_0\}$ 及其 Subspace Polynomial $s_1(X)$

$$
s_1(X) = X \cdot (X + \beta_0) \tag{14}
$$

很显然，$s_1(V_1) = \{0, 0\}$，如果将 $s_1(X)$ 作用到 V_3 ，我们会得到下面的结果：

$$
\begin{aligned}
s_1(0) &= 0 \\
s_1(\beta_0) &= 0 \\
s_1(\beta_1) &= \beta_0\beta_1 + \beta_1^2 \\
s_1(\beta_0 + \beta_1) &= \beta_0\beta_1 + \beta_1^2 \\
s_1(\beta_2) &= \beta_0\beta_2 + \beta_2^2 \\
s_1(\beta_0 + \beta_2) &= \beta_0\beta_2 + \beta_2^2 \\
s_1(\beta_1 + \beta_2) &= \beta_0\beta_1 + \beta_1^2 + \beta_0\beta_2 + \beta_2^2 \\
s_1(\beta_0 + \beta_1 + \beta_2) &= \beta_0\beta_1 + \beta_1^2 + \beta_0\beta_2 + \beta_2^2
\end{aligned} \tag{15}
$$

上面的等式显示 $s_1(V_3)$ 被映射到了一个大小只有 V_3 一半的集合，记为 V_2。该集合也是一个子空间，$V_2 = \langle \beta'_0, \beta'_1 \rangle = \langle \beta_0\beta_1 + \beta_1^2, \beta_0\beta_2 + \beta_2^2 \rangle$，维度为 2。

这不是巧合，根据群同构定理，同态映射 $\phi : H \to G$ 的 Image G 满足 $G \cong H/Ker(\phi)$，其中 G 是一个商群，并且 $|G| = |H|/|Ker(\phi)|$。在上面这个例子里，$s_1 : V_3 \to V_2$ 是同态映射，$V_1 = Ker(s_1)$。

映射构成的链

对于 $V_2 = s_1(V_3)$，我们仍然可以继续构造一个 Degree 为 2 的 Subspace Polynomial，

$$
s'_1(X) = X \cdot (X + \beta_0\beta_1 + \beta_1^2) \tag{16}
$$

可以继续将 V_2 映射到一个一维的子空间 $V_1 = \langle \beta'' \rangle$。我们只需要计算 $s_1(\beta'_0)$ 与 $s_1(\beta'_1)$ 即可，这些 Basis 分量构成了 V_2：

$$
\begin{aligned}
s'_1(\beta_0\beta_1 + \beta_1^2) &= 0 \\
s'_1(\beta_0\beta_1 + \beta_1^2 + \beta_0\beta_2 + \beta_2^2) &= \beta_0^2\beta_1\beta_2 + \beta_0\beta_1^2\beta_2 + \beta_0^2\beta_2^2 + \beta_0\beta_1\beta_2^2 + \beta_1^2\beta_2^2 + \beta_2^4 \\
&= \beta''
\end{aligned} \tag{17}
$$

其中 V_2 的 Basis (β'_0, β'_1) 中第一个分量会被映射到 0，第二个分量映射到 β''。

至此，我们得到一个映射的链：

$$
V_3 \xrightarrow{s_1} V_2 \xrightarrow{s'_1} V_1 \tag{18}
$$

或者也可以写为：

$$
\langle \beta_0, \beta_1, \beta_2 \rangle \xrightarrow{s_1} \langle s_1(\beta_1), s_1(\beta_2) \rangle \xrightarrow{s'_1} \langle s'_1(s_1(\beta_2)) \rangle \tag{19}
$$

并且每次映射到的线性子空间的维度都减一，即集合大小减半。这个代数结构是我们后续构造 FFT 与 FRI 协议的关键。

不难证明，对于任意的线性子空间，只要我们选定一个 Basis 之后，就可以依次构造 Degree 为 2 的 Subspace Polynomial 作为映射函数，然后通过映射得到一个维度减 1 的子空间，然后不断重复，直到子空间降至 1 维。当然 Basis 选择不同，以及 Subspace Polynomial 的选择不同都会导致不同的映射链。选择恰当的映射链可以显著提高计算的效率。

s_1 映射的复合

我们定义映射链的初始子空间为 $S^{(0)}$，映射后的子空间为 $S^{(1)}$，i 次映射之后的子空间为 $S^{(i)}$：

$$
S^{(0)} \xrightarrow{s_1} S^{(1)} \xrightarrow{s_1^{(1)}} \cdots \xrightarrow{s_1^{(n-1)}} S^{(n)} \tag{20}
$$

给定 $S^{(i)}$ 的一组 Basis 后，假设为 $B^{(i)} = (\beta_0^{(i)}, \beta_1^{(i)}, \ldots, \beta_s^{(i)})$，在 Basis 上定义 Subspace Polynomial $s_1^{(i)}$，并用其作为群同态映射函数，把 $S^{(i)}$ 降维到 $S^{(i+1)}$。降维后的线性子空间 $S^{(i+1)}$ 的 Basis 需要把 $S^{(i)}$ 的 Basis 同步跟着 $s_1^{(i)}$ 转换到一个新的 Basis。切换到新 Basis 之后，我们又可以定义一组新的 Subspace Polynomial $s_i^{(i+1)}(X)$。

我们假设 $S^{(0)} = \langle \beta_0, \beta_1, \ldots, \beta_{k-1} \rangle$ 开始，给定一组 Basis B_k，经过 s_1 的映射之后，我们得到了 $S^{(1)}$，以及其 Basis $B^{(1)}$：

$$
B^{(1)} = \langle s_1(\beta_1), s_1(\beta_2), \ldots, s_1(\beta_{k-1}) \rangle \tag{21}
$$

在 $S^{(1)}$ 再定义 $s_1^{(1)}(X)$：

$$s_1^{(1)}(X) = X(X + s_1(\beta_1)) \tag{22}$$

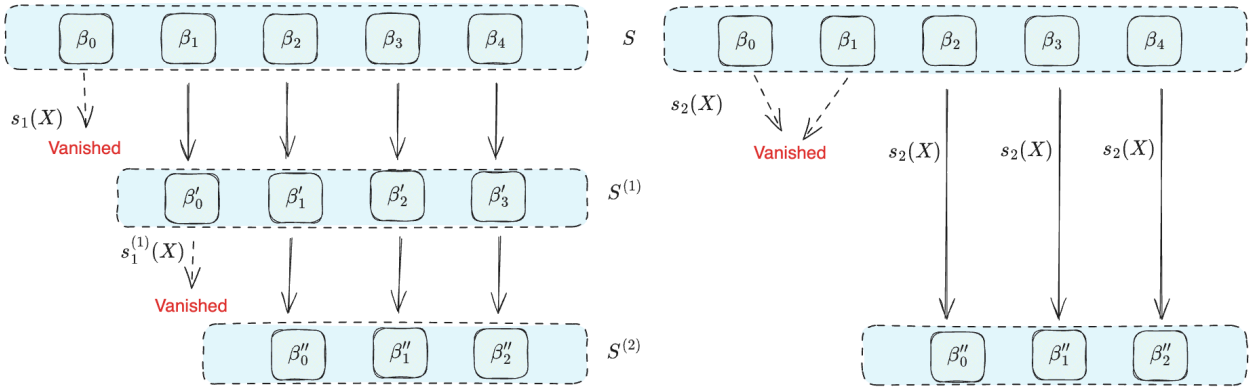
那么，$S^{(1)}$ 映射后产生的 $S^{(2)}$ 和 $S^{(0)}$ 之间的关系是什么？对于任何一个元素 $a \in S^{(0)}$，它先经过 s_1 映射到 $S^{(1)}$，然后再经过 $s_1^{(1)}$ 映射到 $S^{(2)}$ 中的某个元素，因此经过两次映射后的值可以被写为两次映射函数的复合，$s_1^{(1)}(s_1(X))$，我们化简下这个复合函数：

$$\begin{aligned} s_1^{(1)}(s_1(X)) &= s_1(X)(s_1(X) + s_1(\beta_1)) \\ &= s_1(X)(s_1(X + \beta_1)) && (\text{additive homomorphism }) \\ &= s_2(X) && (\text{recurrency }) \end{aligned}$$

于是我们推导出了 $s_1^{(1)}(s_1(X)) = s_2(X)$。这意味着经过两次 2-to-1 的映射，等价于做一次 4-to-1 的映射，并且对应的同态映射函数为 s_2：

$$\begin{aligned} s_2 : & S^{(0)} \to S^{(2)} \\ & X \mapsto X(X + \beta_0)(X + \beta_1)(X + \beta_1 + \beta_0) \end{aligned} \tag{23}$$

如下图所示，左右两种映射方式都会得到 $S^{(2)}$：



同理，我们可以得到下面的结论，对于折叠 j 次之后的线性子空间 $S^{(j)}$ 为

$$S^{(j)} = \langle s_j(\beta_j), s_j(\beta_{j+1}), \ldots, s_j(\beta_{k-1}) \rangle \tag{24}$$

并且 s_j 满足下面的复合等式：

$$s_j(X) = s_{j-1}^{(1)}(s_1(X)) = s_{j-1}^{(1)} \circ s_1 \tag{25}$$

这个复合映射的等式可以解读为：先做一次 s_1 映射得到 $S^{(1)}$，然后再做一次 $j-1$ 维的映射 $s_{j-1}^{(1)}$，等价于直接一次 j 维映射 s_j，两者都映射到同一个子空间 $S^{(j)}$。

同样，我们还能证明：如果先做一次 $j-1$ 维的映射 s_{j-1}，再在映射后的子空间 $S^{(j-1)}$ 上做一次一维映射，同样可以得到子空间 $S^{(j)}$：

$$s_j(X) = s_1^{(1)}(s_{j-1}(X)) = s_1^{(1)} \circ s_{j-1} \tag{26}$$

更一般地，我们可以证明得到下面的重要性质，即在任意子空间 $S^{(i)}$ 上做一次 j 维映射，等价于在其上连续做 j 次 1 维映射：

$$s_j^{(i)}(X) = s_1^{(i+j-1)} \circ s_1^{(i+j-2)} \circ \cdots \circ s_1^{(i)} \tag{27}$$

多项式的 Polynomial Basis

对于一个次数小于 $N = 2^n$ 的一元多项式 $f(X) \in \mathbb{F}[X]^{<N}$，它有两种常见的表达形式，「系数式」与「点值式」。其中系数式是我们最常见的形式：

$$f(X) = c_0 + c_1 X + c_2 X^2 + \ldots + c_{N-1} X^{N-1} \tag{28}$$

其中 $\vec{c} = (c_0, c_1, \ldots, c_{N-1})$ 为多项式的系数向量。另外未知数的向量 $(1, X, X^2, \ldots, X^{N-1})$ 则构成了一组多项式的基（Basis），按惯例称之为 Monomial Basis，记为 $\mathcal{B}^{mono}$：

$$\mathcal{B}^{mono} = (1, X, X^2, \ldots, X^{N-1}) \tag{29}$$

这个基向量也可以表达为下面的 Tensor Product 形式：

$$\mathcal{B}^{mono} = (1, X) \otimes (1, X^2) \otimes \ldots \otimes (1, X^{2^{n-1}}) \tag{30}$$

一元多项式的 「点值式」称为 Lagrange Basis 表示。即我们可以用 N 个「系数」来唯一确定一个 Degree 小于 N 的多项式（请注意，这里的系数是更广泛意义上的概念，而不是局限于「系数式」表示中的系数）。

通过多项式除法，我们可以得到多项式在 $\mathcal{B}^{mono}$ 上的系数。例如有一个 degree 为 7 的多项式 $t(X)$，那么我们可以先计算 $X^4 \cdot X^2 \cdot X$ 的系数，即计算多项式除法：$t(X) / (X^4 \cdot X^2 \cdot X)$，得到一个系数 c_7 与一个余数多项式 $t'(X)$；然后再计算 $t'(X) / (X^4 \cdot X^2)$，得到 $\mathcal{B}_6^{mono} = X^6$ 的系数 c_6，以此类推，最终我们可以得到 $t(X)$ 关于 $\mathcal{B}^{mono}$ 的系数向量 $\vec{c} = (c_0, c_1, \ldots, c_7)$，使得：

$$t(X) = c_0 + c_1 X + c_2 X^2 + \ldots + c_7 X^7 \tag{31}$$

利用前文已讨论过的 Subspace Polynomial $s_k(X)$，我们可以定义一组新的 Basis。根据其定义，$s_k(X)$ 的 degree 恰好也是 2^k，类似 $(1, X, X^2, X^4)$，因此 $(s_0(X), s_1(X), s_2(X))$ 也可作为构造多项式 Basis 的基本原料。仿照 $\mathcal{B}^{mono}$ 的定义，我们定义 (Novel) Polynomial Basis $\mathcal{B}^{novel}$：

$$\mathcal{B}^{novel} = (1, s_0(X)) \otimes (1, s_1(X)) \otimes \ldots \otimes (1, s_{n-1}(X)) \tag{32}$$

请注意与论文 [LCH14] 和 [DP24] 不同的是，这里我们暂时没有引入 Normalized Subspace Polynomial，以方便大家理解。回到上面的定义，其中每一个分量 $\mathcal{B}_i^{novel}$ 我们简记为 $\mathcal{X}_i(X)$，定义如下：

$$\mathcal{X}_i(X) = \prod_{j=0}^{n-1} (s_j(X))^{i_j}, \text{ where } \mathsf{bits}(i) = (i_0, i_1, \ldots, i_{n-1}) \tag{33}$$

这里 $\mathsf{bits}(i)$ 表示把整数 i 按照二进制展开，比如 $i = 5$，那么 $\mathsf{bits}(5) = (1, 0, 1)$，$\mathsf{bits}(6) = (0, 1, 1)$。举例 $n = 3, N = 8$，按照上面的定义，我们可以计算出一组多项式基 $\big(\mathcal{X}_0(X), \mathcal{X}_1(X), \ldots, \mathcal{X}_7(X)\big)$

$$\begin{aligned}
\mathcal{X}_0(X) &= 1 \\
\mathcal{X}_1(X) &= s_0(X) = X \\
\mathcal{X}_2(X) &= s_1(X) \\
\mathcal{X}_3(X) &= s_0(X) \cdot s_1(X) \\
\mathcal{X}_4(X) &= s_2(X) \\
\mathcal{X}_5(X) &= s_0(X) \cdot s_2(X) \\
\mathcal{X}_6(X) &= s_1(X) \cdot s_2(X) \\
\mathcal{X}_7(X) &= s_0(X) \cdot s_1(X) \cdot s_2(X)
\end{aligned} \tag{34}$$

容易检验，每一个 Basis 分量 $\mathcal{X}_i(X)$ 的 Degree 恰好为 i，因此 $\mathcal{B}^{novel}$ 就构成了一组线性无关的多项式 Basis。对于任意的 Degree 小于 8 的多项式 $f(X) \in \mathbb{F}_{2^m}[X]$：

$$\begin{aligned}
f(X) &= a_0 \mathcal{X}_0(X) + a_1 \mathcal{X}_1(X) + \ldots + a_7 \mathcal{X}_7(X) \\
&= a_0 + a_1 s_0(X) + a_2 s_1(X) + a_3 s_0(X) \cdot s_1(X) \\
&+ a_4 s_2(X) + a_5 s_0(X) \cdot s_2(X) + a_6 s_1(X) \cdot s_2(X) + a_7 s_0(X) \cdot s_1(X) \cdot s_2(X)
\end{aligned} \tag{35}$$

同样，我们可以利用多项式除法来将一个多项式在 $\mathcal{B}^{novel}$ 和 $\mathcal{B}^{mono}$ 之间转换。

Additive FFT

类似 Multiplicative FFT，要构造 Additive FFT，我们需要在 $\mathbb{F}_{2^m}$ 中定义一个加法子群的映射链。如前所述，Subspace Polynomials 恰好可以用来构造这个映射链。同时 Subspace Polynomials 又可以构造一组多项式 Basis。

$$S^{(0)} \xrightarrow{s_1} S^{(1)} \xrightarrow{s_1^{(1)}} \cdots \xrightarrow{s_1^{(n-1)}} S^{(n)} \tag{36}$$

为了演示方便，指定 $n=3$，$S^{(0)} = \langle \beta_0, \beta_1, \beta_2 \rangle$ 。仿照 Multiplicative FFT 的思路，我们将用 $\mathcal{B}^{novel}$ 表示的多项式 $f(X)$（Degree 为 7）进行奇偶项拆分，拆分成两个次数减半的多项式：

$$\begin{aligned}
f(X) &= a_0\mathcal{X}_0(X) + a_1\mathcal{X}_1(X) + \ldots + a_7\mathcal{X}_7(X) \\
&= a_0 + a_1 s_0(X) + a_2 s_1(X) + a_3 s_0(X)\cdot s_1(X) \\
&\quad + a_4 s_2(X) + a_5 s_0(X)\cdot s_2(X) + a_6 s_1(X)\cdot s_2(X) + a_7 s_0(X)\cdot s_1(X)\cdot s_2(X) \\
&= \big(a_0 + a_2 s_1(X) + a_4 s_2(X) + a_6 s_1(X)\cdot s_2(X)\big) \\
&\quad + \big(a_1 + a_3 s_0(X)\cdot s_1(X) + a_5 s_0(X)\cdot s_2(X) + a_7 s_0(X)\cdot s_1(X)\cdot s_2(X)\big) \\
&= \big(a_0 + a_2 s_1(X) + a_4 s_2(X) + a_6 s_1(X)\cdot s_2(X)\big) \\
&\quad + s_0(X)\cdot\big(a_1 + a_3\cdot s_1(X) + a_5\cdot s_2(X) + a_7\cdot s_1(X)\cdot s_2(X)\big)
\end{aligned} \tag{37}$$

然后我们引入两个辅助多项式 $f_{even}(X), f_{odd}(X)$ ，它们

$$\begin{aligned}
f_{even}(X) &= a_0 + a_2\cdot s_1(X) + a_4\cdot s_2(X) + a_6\cdot s_1(X)\cdot s_2(X) \\
f_{odd}(X) &= a_1 + a_3\cdot s_1(X) + a_5\cdot s_2(X) + a_7\cdot s_1(X)\cdot s_2(X)
\end{aligned} \tag{38}$$

根据我们之前推导的映射的复合性质，$s_1(X) = s_0^{(1)} \circ s_0(X)$，$s_2(X) = s_1^{(1)} \circ s_1(X)$，于是我们可以得到：

$$\begin{aligned}
f_{even}(X) &= a_0 + a_2\cdot s_0^{(1)}(s_1(X)) + a_4\cdot s_1^{(1)}(s_1(X)) + a_6\cdot s_0^{(1)}(s_1(X))\cdot s_1^{(1)}(s_1(X)) \\
&= a_0 + a_2\cdot s_0^{(1)}(\color{blue}{s_1(X)}) + a_4\cdot s_1^{(1)}(\color{blue}{s_1(X)}) + a_6\cdot s_0^{(1)}(\color{blue}{s_1(X)})\cdot s_1^{(1)}(\color{blue}{s_1(X)}) \\
f_{odd}(X) &= a_1 + a_3\cdot s_0^{(1)}(s_1(X)) + a_5\cdot s_1^{(1)}(s_1(X)) + a_7\cdot s_0^{(1)}(s_1(X))\cdot s_1^{(1)}(s_1(X)) \\
&= a_1 + a_3\cdot s_0^{(1)}(\color{blue}{s_1(X)}) + a_5\cdot s_1^{(1)}(\color{blue}{s_1(X)}) + a_7\cdot s_0^{(1)}(\color{blue}{s_1(X)})\cdot s_1^{(1)}(\color{blue}{s_1(X)})
\end{aligned} \tag{39}$$

代入 $\color{blue}{Y = s_1(X)}$ 后，我们可以把 $f(X)$ 拆分成关于 $f_{even}(\color{blue}{Y})$ 和 $f_{odd}(\color{blue}{Y})$ 的等式：

$$f(X) = f_{even}(\color{blue}{Y}) + s_0(X)\cdot f_{odd}(\color{blue}{Y}) \tag{40}$$

而多项式 $f_{even}(\color{blue}{Y})$ 和 $f_{odd}(\color{blue}{Y})$ 正好是定义在 $\mathcal{X}^{(1)}$ 上的多项式：

$$\begin{aligned}
\mathcal{X}_0^{(1)}(X) &= 1 \\
\mathcal{X}_1^{(1)}(X) &= s_0^{(1)}(X) &&= s_0(s_1(X)) &&= s_1(X) \\
\mathcal{X}_2^{(1)}(X) &= s_1^{(1)}(X) &&= s_1(s_1(X)) &&= s_2(X) \\
\mathcal{X}_3^{(1)}(X) &= s_0^{(1)}(X)\cdot s_1^{(1)}(X) &&= s_0(s_1(X))\cdot s_1(s_1(X)) &&= s_1(X)\cdot s_2(X)
\end{aligned} \tag{41}$$

重写下奇偶多项式：

$$\begin{aligned}
f_{even}(X) &= a_0\cdot\mathcal{X}_0^{(1)}(X) + a_2\cdot\mathcal{X}_1^{(1)}(X) + a_4\cdot\mathcal{X}_2^{(1)}(X) + a_6\cdot\mathcal{X}_3^{(1)}(X) \\
f_{odd}(X) &= a_1\cdot\mathcal{X}_0^{(1)}(X) + a_3\cdot\mathcal{X}_1^{(1)}(X) + a_5\cdot\mathcal{X}_2^{(1)}(X) + a_7\cdot\mathcal{X}_3^{(1)}(X)
\end{aligned} \tag{42}$$

从结构上看，这个等式与 Multiplicative FFT 中的 $f(X) = f_{even}(X^2) + X\cdot f_{odd}(X^2)$ 拆分非常相似；而 $X \mapsto X^2$ 映射也对应于 $s_1 : X \mapsto X(X+\beta_0)$ 映射。而 $S^{(0)}$ 在 s_1 的映射下，产生出一个尺寸只有原来一半的子空间 $S^{(1)}$：

$$S^{(1)} = \langle s_1(\beta_1), s_1(\beta_2) \rangle \tag{43}$$

于是我们可以依赖递归调用，求得 $\{f_{even}(X) \mid X \in S^{(1)}\}$ 与 $\{f_{odd}(X) \mid X \in S^{(1)}\}$，然后再利用 $f(X) = f_{even}(X) + s_0(X)\cdot f_{odd}(X)$ 这个等式得到 $f(X)$ 在 $S^{(0)}$ 上的值。

下面我们假设递归调用成功返回，那么我们就得到了 $f_{even}(X)$ 和 $f_{odd}(X)$ 在 $S^{(1)}$ 上的全部求值，记为 $\vec{u}$ 与 $\vec{v}$，定义如下：

$$
\begin{aligned}
(u_0, u_1, u_2, u_3) &= \big(f_{even}(0), f_{even}(1), f_{even}(s_1(\beta_1)), f_{even}(s_1(\beta_1)+1)\big) \\
(v_0, v_1, v_2, v_3) &= \big(f_{odd}(0), f_{odd}(1), f_{odd}(s_1(\beta_1)), f_{odd}(s_1(\beta_1)+1)\big)
\end{aligned} \tag{44}
$$

然后我们就可以计算 $f(X)$ 在 $S^{(0)}$ 上的全部求值，即 $f(X)\mid_{S^{(0)}}$：

$$
\begin{aligned}
f(0) &= f_{even}(s_1(0)) + 0 \cdot f_{odd}(s_1(0)) \\
&= u_0 \\
f(1) &= f_{even}(s_1(1)) + 1 \cdot v_1 \\
&= u_0 + v_1 \\
f(\beta_1) &= f_{even}(s_1(\beta_1)) + \beta_1 \cdot f_{odd}(s_1(\beta_1)) \\
&= u_1 + \beta_1 \cdot v_1 \\
f(\beta_1+1) &= f_{even}(s_1(\beta_1)+s_1(1)) + (\beta_1+1) \cdot f_{odd}(s_1(\beta_1)+s_1(1)) \\
&= u_1 + \beta_1 \cdot v_1 + v_1 \\
f(\beta_2) &= f_{even}(s_1(\beta_2)) + \beta_2 \cdot f_{odd}(s_1(\beta_2)) \\
&= u_2 + \beta_2 \cdot v_2 \\
f(\beta_2+1) &= f_{even}(s_1(\beta_2)+s_1(1)) + (\beta_2+1) \cdot f_{odd}(s_1(\beta_2)+s_1(1)) \\
&= u_2 + \beta_2 \cdot v_2 + v_2 \\
f(\beta_2+\beta_1) &= f_{even}(s_1(\beta_2)+s_1(\beta_1)) + (\beta_2+\beta_1) \cdot f_{odd}(s_1(\beta_2)+s_1(\beta_1)) \\
&= u_3 + \beta_2 \cdot v_3 + \beta_1 \cdot v_3 \\
f(\beta_2+\beta_1+1) &= f_{even}(s_1(\beta_2)+s_1(\beta_1)+s_1(1)) + (\beta_2+\beta_1+1) \cdot f_{odd}(s_1(\beta_2)+s_1(\beta_1)+s_1(1)) \\
&= u_3 + \beta_2 \cdot v_3 + \beta_1 \cdot v_3 + v_3
\end{aligned} \tag{45}
$$

我们把上面这个 Additive FFT 递归算法用 Python 代码实现如下：

```python
def afft(f, k, B):
    """
    Perform the Additive Fast Fourier Transform (AFFT) on a given polynomial.

    Args:
        f (list): Coefficients of the polynomial to be transformed.
        k (int): The depth of recursion, where 2^k is the size of the domain.
        B (list): The basis of the domain over which the polynomial is evaluated.

    Returns:
        list: The evaluations of the polynomial over the domain.
    """
    if k == 0:
        return [f[0]]
    half = 2**(k-1)

    f_even = f[::2]
    f_odd = f[1::2]

    V = span(B)                                 # the subspace spanned by B
    q = lambda x: x*(x+B[0])/(B[1]*(B[1] + 1)) # s^(i)_1 map
    B_half = [q(b) for b in B[1:]]              # the basis of the mapped subspace

    e_even = afft(f_even, k-1, B_half)  # compute the evaluations of f_even
    e_odd  = afft(f_odd, k-1, B_half)   # compute the evaluations of f_odd

    e = [0] * (2 * half)                # initialize the list of evaluations
```

```
    for i in range(0, half):
        e[2*i]   = e_even[i] + V[2*i] * e_odd[i]
        e[2*i+1] = e_even[i] + V[2*i+1] * e_odd[i]

    return e
```

函数 `afft(f, k, B)` 总共有三个参数，分别是多项式 $f(X)$ 在 $\mathcal{B}^{novel}$ 上的系数向量，递归深度 k，以及当前的子空间 $S^{(0)}$ 的 Basis。

总结

Additive FFT 算法需要一个通过 Subspace Polynomial 构造的子空间的映射链。本文介绍的原理并不局限在递归构造的二进制域，而是一种更广泛的代数结构。 在 [LCH14] 论文中采用的是另外一种递归 Additive FFT 算法，我们将在下一篇文中介绍两者的差异，以及 [DP24] 论文中的 Additive FFT 迭代算法（Algorithm 2）。

References

- [DP24] Benjamin E. Diamond and Jim Posen. "Polylogarithmic Proofs for Multilinears over Binary Towers". 2024. https://eprint.iacr.org/2024/504
- [LCH14] Lin, Sian-Jheng, Wei-Ho Chung, and Yunghsiang S. Han. "Novel polynomial basis and its application to Reed-Solomon erasure codes." 2014 ieee 55th annual symposium on foundations of computer science. IEEE, 2014. https://arxiv.org/abs/1404.3458
- [LN97] Lidl, Rudolf, and Harald Niederreiter. Finite fields. No. 20. Cambridge university press, 1997.