August 6th 2024

Introduction to Additive NTT

Recall that NTT over a large finite field $\mathbb{F}_p$ works by transforming monomial basis coeffs of a univariate polynomial to Lagrange Basis coefficients over a carefully selected domain (read: evals of said polynomial over some input domain D)

- Suppose the transform is for univariate polynomials of degree $< 2^l$, i.e. $\mathbb{F}_p[X]^{<2^l}$
- let $f(x) \in \mathbb{F}_p[X]^{<2^l} = a_0 + a_1 X + \cdots + a_{2^l-1} X^{2^l-1}$
- NTT transforms $\vec{a} = (a_i)_{i=0,\ldots,2^l-1}$ into $\vec{b} = (b_i)_{i=0,\ldots,2^l-1}$

  where $b_i = f(d_i)$ and $D = \{d_i \mid i \in \{0,\ldots,2^l-1\}\} \subseteq \mathbb{F}_p$ is the carefully selected domain
- when $D = \langle w_{2^l} \rangle$, i.e. the (multiplicative) subgroup of $\mathbb{F}_p$ generated by the 2^l-th root of unity, traditional NTT and iNTT admit, respectively, $\Theta(l \cdot 2^l)$ algorithms to perform change of basis from $\vec{a}$ to $\vec{b}$ and $\vec{b}$ to $\vec{a}$.
- For binary fields, however, this 2^l-th root of unity will not exist, and we would like to find a new way to perform a change of basis in time $\Theta(l \cdot 2^l)$

Enter Additive NTT

Outline

1. Subspace Polynomials
2. Novel Polynomial Basis
3. Recursive Structure
3.5. Algorithmic Structure
4. The forward transform
5. Linear Codes
6. Reed Solomon Encoding
7. Quick Note on inverse transform and RS Decoding

① Subspace Polynomials

- Fix a binary field L of degree r over $\mathbb{F}_2$ (i.e. $L \cong \mathbb{F}_{2^r}$)
- Fix $\beta = (\beta_0, \dots, \beta_{r-1})$ as an r-dimensional $\mathbb{F}_2$-basis for L
  ($\beta_i \in L$, β_i's are $\mathbb{F}_2$-linearly independent)
- Let $U_i = \langle \beta_0, \dots, \beta_{i-1} \rangle$ be the subspace of L generated by the first i basis vectors
  ↖ span of $\beta_0, \dots, \beta_{i-1}$
  for $i = 0, \dots, r$
- Defn: A subspace polynomial $W(X) \in L[X]$ splits completely over L and its roots, each of multiplicity 1, form an $\mathbb{F}_2$-linear subspace of L
- Define $W_i(X) := \prod_{u \in U_i} X + u$ and $\hat{W}_i(X) := \frac{W_i(X)}{W_i(\beta_i)}$
  → The roots of $W_i(X)$ are U_i
  ↖ normalized variant. $\hat{W}_i(\beta_i) = 1$
- Note: $W_0(X) := \prod_{u \in U_0} X + u = X + 0 = X$
- Note: $\deg(W_i(X)) = 2^i$

Lemma

$$U_{i+1} = U_i \cup \{ u + \beta_i \mid u \in U_i \}$$

- $W_{i+1}(x) = W_i(x) \cdot W_i(x + \beta_i)$ ✡

$$\prod_{u \in U_{i+1}} X + u = \left[\prod_{u \in U_i} x + u \right] \cdot \left[\prod_{u \in U_i} x + (\beta_i + u) \right]$$ by lemma

$$= W_i(x) \cdot [W_i(x) + W_i(\beta_i)]$$

- For $b \in \{0, \dots, 2^r - 1\}$, let $b_0, \dots, b_{r-1}$ denote its bits such that $b = \sum_{i=0}^{r-1} b_i \cdot 2^i$

  Then we denote as $w_b = [\beta_0, \beta_1, \dots, \beta_{r-1}] \cdot [b_0, b_1, \dots, b_{r-1}]$
  (basis elements; $\mathbb{F}_2$-coeffs)

  $= b_0\beta_0 + b_1\beta_1 + \dots + b_{r-1}\beta_{r-1}$

  Thus $L = \{w_b \mid b \in \{0, \dots, 2^r - 1\}\}$

- Note: $\beta_i = w_{2^i}$

Summary of important results related to subspace polynomial

- $\forall x, y \in L$, $\forall K \in \{0, \dots, r-1\}$

$$W_K(x+y) = W_K(x) + W_K(y)$$

$$\hat{W}_K(x+y) = \hat{W}_K(x) + \hat{W}_K(y)$$

- $\forall K \in \{0, \dots, r-1\}$, $\forall b \in \{0, \dots, 2^K - 1\}$

$$\left.\begin{aligned} W_K(w_b) &= 0 \\ \hat{W}_K(w_b) &= 0 \end{aligned}\right| \; w_b \in U_K \Rightarrow W_K(w_b)$$

- $\forall K \in \{0, \dots, r-2\}$

$$W_{K+1}(x) = W_K(x) \cdot W_K(x + \beta_K)$$

(2) Novel Polynomial Basis

- We have fixed L as a degree r extension of $\mathbb{F}_2$ with basis $\beta = (\beta_0, \dots, \beta_{r-1})$
- Now, fix $\ell \in \{0, \dots, r-1\}$
- We denote as $L[X]^{<2^\ell}$ the ring of polynomials with L-coefficients of degree $< 2^\ell$.

$$L[X]^{<2^\ell} = \left\{ P(X) \in L[X] \;\middle|\; \deg(P) < 2^\ell \right\}$$

- $L[X]^{<2^\ell}$ is a 2^ℓ-dimensional vector space over L, and therefore has a basis of size 2^ℓ.
- A natural basis would be the (univariate) monomial basis $(1, X, X^2, \dots, X^{2^\ell - 1})$
  However, we do not know a $O(\ell \cdot 2^\ell)$ algorithm to change basis btwn monomial and a Lagrange basis (i.e. we do not have a $O(\ell \cdot 2^\ell)$ algo that gives us 2^ℓ unique evaluations given monomial coeffs as input)
- Define $X_i(x) = \prod_{k=0}^{\ell-1} \left(\hat{W}_k(x)\right)^{i_k} \quad \forall i \in \{0, \dots, 2^\ell - 1\}$

  where
  - $i_k \in \{0, 1\}$
  - $i = \sum_{k=0}^{\ell-1} i_k \cdot 2^k$
- Note $\deg(X_i(x)) = i$
- Now $(X_0(x), \dots, X_{2^\ell - 1}(x))$ admits an L-basis for $L[X]^{<2^\ell}$ called the NOVEL POLYNOMIAL BASIS

(3) Recursive Structure of Novel Polynomial Basis

- We have fixed binary field L of degree r over $\mathbb{F}_2$ with basis $\beta = (\beta_0, \dots, \beta_{r-1})$
- We have fixed $\ell \in \{0, \dots, r-1\}$
- Fix $D(x) = d_0 X_0(x) + \cdots + d_{2^\ell - 1} X_{2^\ell - 1}(x) \in L[X]^{<2^\ell}$
- Define $\Delta_\ell^m(x) = d_m \quad \forall m \in \{0, \dots, 2^\ell - 1\}$

$$\Delta_i^m(x) = \Delta_{i+1}^m(x) + \hat{W}_i(x) \cdot \Delta_{i+1}^{m+2^i}(x)$$

$$\forall i \in \{0, \dots, \ell-1\} \quad \forall m \in \{0, \dots, 2^i - 1\}$$

- $\deg_x\left(\Delta_i^m(x)\right) \le 2^{\ell-i} - 1 \quad \forall m \in \{0, \dots, 2^i - 1\}$
- Claim without proof: $\Delta_0^0(x) = D(x)$

ex $\ell = 3$, $D(x) = \sum_{i=0}^{7} d_i X_i(x)$

$$= [d_0, \dots, d_7] \cdot [X_0(x), \dots, X_7(x)]$$

$$= \left[d_0 + d_4 \hat{W}_2(x) + \hat{W}_1(x)\left[d_2 + d_6 \hat{W}_2(x)\right]\right]$$

$$+ \hat{W}_0(x)\left[d_1 + d_5 \hat{W}_2(x) + \hat{W}_1(x)\left[d_3 + d_7 \hat{W}_2(x)\right]\right]$$

$$= \left[\Delta_2^0(x) + \hat{W}_1(x)\Delta_2^2(x)\right]$$

$$+ \hat{W}_0(x)\left[\Delta_2^1(x) + \hat{W}_1(x)\Delta_2^3(x)\right]$$

$$= \Delta_1^0(x) + \hat{W}_0(x)\Delta_1^1(x) = \Delta_0^0(x) \qquad \square$$

- **Lemma:** $\Delta_i^m(x+y) = \Delta_i^m(x) \quad \forall y \in \{w_b\}_{b=0}^{2^i-1}$

- **Proof** By induction on i

BC $i = \ell - 1$ $\quad \Delta_{\ell-1}^m(x)$

$$= \Delta_\ell^m(x) + \hat{w}_{\ell-1}(x)\,\Delta_\ell^{m+2^{\ell-1}}(x)$$

$$= d_m + \hat{w}_{\ell-1}(x)\,d_{m+2^{\ell-1}}$$

similarly $\Delta_{\ell-1}^m(x+y) = d_m + \hat{w}_{\ell-1}(x+y)\,d_{m+2^{\ell-1}}$

but $\hat{w}_{\ell-1}(x+y) = \hat{w}_{\ell-1}(x)$ when $y \in \{w_b\}_{b=0}^{2^{\ell-1}-1}$

IH Assume true for $i = c+1$

IS $\Delta_c^m(x+y)$

$$= \Delta_{c+1}^m(x+y) + \hat{w}_c(x+y)\,\Delta_{c+1}^{m+2^c}(x+y)$$

$$= \Delta_{c+1}^m(x) + \hat{w}_c(x)\,\Delta_{c+1}^{m+2^c}(x)$$ // by IH and $\hat{w}(x+y) = \hat{w}(x)$

3.5 Algorithmic Structure

- We have fixed binary field L of degree r over $\mathbb{F}_2$ with basis $\vec{\beta} = (\beta_0, \dots, \beta_{r-1})$
- We have fixed a parameter $\ell \in \{0, \dots, r-1\}$ and consider the 2^ℓ-dimensional L-vector space $L[X]^{<2^\ell}$ (univariate polys over L with degree $< 2^\ell$)
- we have fixed $D(x) = \sum_{i=0}^{2^{\ell}-1} d_i X_i(x) \in L[X]^{<2^\ell}$

Our objective now is to exploit the recursive structure of the novel polynomial basis to build an algorithmic structure for our goal of efficient change-of-basis

- Fix a "shift parameter" $s \in \{0, \dots, 2^r - 1\}$ // more on this later, if easier imagine $s = 0$

Defn

$$\forall i \in \{0, \dots, \ell-1\} \quad \forall m \in \{0, \dots, 2^i - 1\}$$

$$\Psi(i, m, s) := \left\{ \Delta_i^m(w_c + w_s) \;\middle|\; c \in \{b \cdot 2^i\}_{b=0}^{2^{\ell-i}-1} \right\}$$

is a set of size $2^{\ell - i}$

Defn

$$\Psi(\ell, m, s) := \{d_m\}$$ is a singleton set

size $= 2^{\ell-\ell} = 2^0 = 1$

Our goal is to transform $\Psi(\ell, 0, s), \dots, \Psi(\ell, 2^\ell - 1, s)$ to $\Psi(0, 0, s)$ and vice-versa

$\Psi(\ell, 0, s) \to \{d_0\}$, $\Psi(\ell, 2^\ell - 1, s) \to \{d_{2^\ell - 1}\}$, $\Psi(0, 0, s) \to \{\Delta_0^0(w_0), \Delta_0^0(w_1), \dots, \Delta_0^0(w_{2^\ell - 1})\}$ when $s = 0$

- Fix $i \in \{0, \dots, \ell - 1\}$

We will now show how you can relate

$$\forall m \in \{0, \dots, 2^i - 1\} \quad \Psi(i, m, s) \longleftrightarrow \forall m \in \{0, \dots, 2^{i+1} - 1\} \quad \Psi(i+1, m, s)$$

- Fix $m \in \{0, \dots, 2^i - 1\}$

$$\Psi(i,m,s) = \Psi^0(i,m,s) \cup \Psi^1(i,m,s)$$

where

$$\Psi^0(i,m,s) = \left\{ \Delta_i^m(w_c + w_s) \;\middle|\; c \in \{b \cdot 2^i\}_{b=0}^{2^{l-i-1}-1} \right\}$$

$$\Psi^1(i,m,s) = \left\{ \Delta_i^m(w_c + w_{2^i} + w_s) \;\middle|\; c \in \{b \cdot 2^i\}_{b=0}^{2^{l-i-1}-1} \right\}$$

- Fix $b \in \{0, \dots, 2^{l-i-1} - 1\}$ let $c = b \cdot 2^i$

$$\Psi^0(i,m,s)[b] = \Delta_i^m(w_c + w_s) = \underbrace{\Delta_{i+1}^m(w_c + w_s)}_{= \Psi(i+1,m,s)[b]} + \hat{W}_i(w_c + w_s) \underbrace{\Delta_{i+1}^{m+2^i}(w_c + w_s)}_{= \Psi(i+1,m+2^i,s)[b]}$$

$$\begin{aligned}
\Psi^1(i,m,s)[b] = \Delta_i^m(w_c + w_{2^i} + w_s) &= \Delta_{i+1}^m(w_c + w_{2^i} + w_s) + \hat{W}_i(w_c + w_{2^i} + w_s)\Delta_{i+1}^{m+2^i}(w_c + w_{2^i} + w_s) \\
&= \Delta_{i+1}^m(w_c + w_s) + \hat{W}_i(w_c + w_{2^i} + w_s)\Delta_{i+1}^{m+2^i}(w_c + w_s) \\
&= \Delta_{i+1}^m(w_c + w_s) + \left(\hat{W}_i(w_c + w_s) + \hat{W}_i(\beta_i)\right)\Delta_{i+1}^{m+2^i}(w_c + w_s) \\
&= \Delta_{i+1}^m(w_c + w_s) + \left(\hat{W}_i(w_c + w_s) + 1\right)\Delta_{i+1}^{m+2^i}(w_c + w_s) \\
&= \Delta_i^m(w_c + w_s) + \underbrace{\Delta_{i+1}^{m+2^i}(w_c + w_s)}_{= \Psi(i+1,m+2^i,s)[b]}
\end{aligned}$$

- Summary: Fix $i \in \{0, \dots, l-1\}$

For each $m \in \{0, \dots, 2^i - 1\}$ and $b \in \{0, \dots, 2^{l-i-1}\}$

$(\Psi[i,m,s][b], \Psi[i,m,s][b + 2^{l-i-1}])$ can be derived from $(\Psi[i+1,m,s][b], \Psi[i+1,m+2^i,s][b])$

x_0 x_1 (also vice versa) y_0 y_1

The process to convert from (y_0, y_1) into (x_0, x_1) structurally looks like

$$x_0 = y_0 + \square \cdot y_1$$

$$x_1 = x_0 + y_1$$

where $\square$ is some factor depending on i, b that's called in our codebase the "twiddle" factor

$$\square_{i,b} = \hat{W}_i \left(w_s + w_{b \cdot 2^i} \right)$$

(4) The forward transform:

- We have fixed
  - binary field L of degree r over $\mathbb{F}_2$ with basis $\beta = (\beta_0, \ldots, \beta_{r-1})$
  - Parameter $\ell \in \{0, \ldots, r-1\}$ defining our polynomial ring $L[X]_{<2^\ell}$
  - $D(x) = \sum_{k=0}^{2^\ell - 1} d_k \cdot X_k(x) \in L[x]_{<2^\ell}$

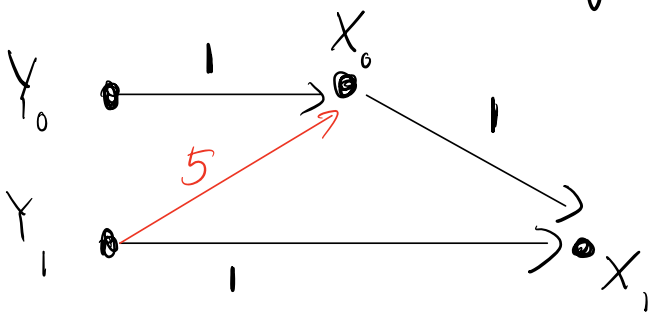
Our goal is to take as input $d = [d_0, \ldots, d_{2^\ell - 1}]$, $s \in \{0, \ldots, 2^r - 1\}$

and output $[D(w_0 + w_s), \ldots, D(w_{2^\ell - 1} + w_s)]$

in time $O(\ell \cdot 2^\ell)$

Let us now introduce the "butterfly" diagram

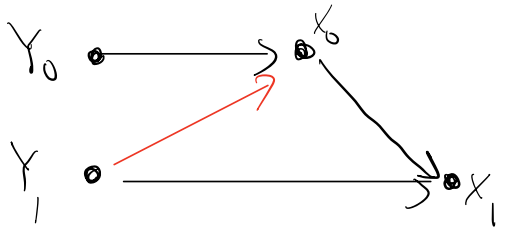
Def:



means $X_0 := 1 \cdot Y_0 + 5 \cdot Y_1$

$X_1 := 1 \cdot X_0 + 1 \cdot Y_1$

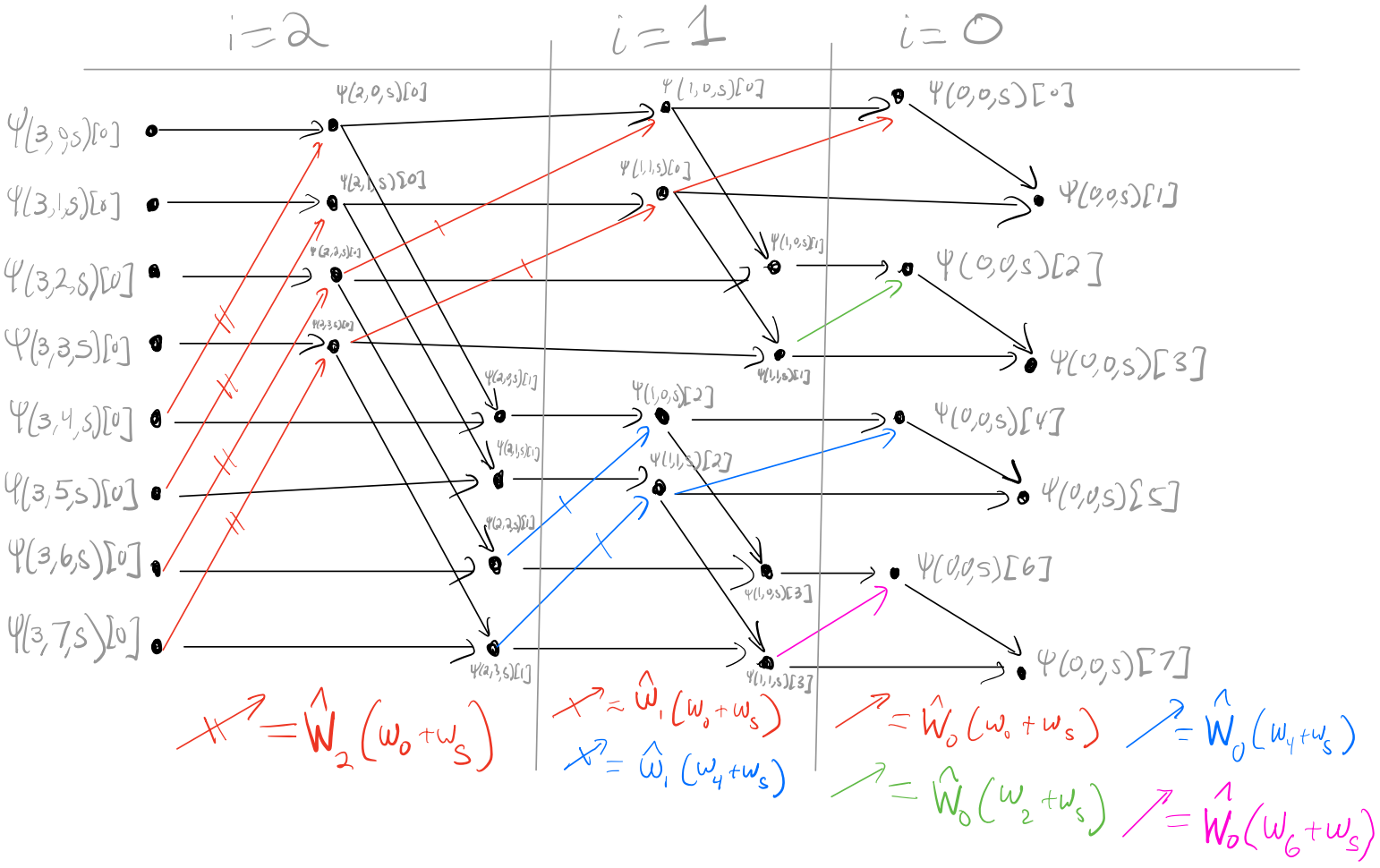
Notes:
- This is a DAG, so one can top-sort and assign nodes values in that order
- A node's value is the sum of its edge-weighted in-node values (weight is multiplicative factor)
- If the edge-weight is 1, we omit explicitly writing this
- Sometimes, we use colors to indicate edge-weights



where $\nearrow = 5$

is the same diagram

ex // Forward transform when $\ell = 3$, arbitrary S



The diagram above suffices as a complete description of the algorithm

Let's talk about how to compute the "twiddle-factors" i.e. $\hat{W}_i(w_{j * block\_size} + w_s)$

- for simplicity, let's assume $S = 2^\ell$, so, $w_s = w_{2^\ell} = \beta_\ell$
- Precompute constants $W_0(\beta_0), \dots, W_0(\beta_{\ell-1}), W_0(\beta_\ell)$
- For $i = 1, \dots, \ell - 1$;
  - Compute $W_i(\beta_i), \dots, W_i(\beta_\ell)$

    using $W_i(\beta_j) := W_{i-1}(\beta_{j-1}) \cdot \left[W_{i-1}(\beta_{j-1}) + W_{i-1}(\beta_{i-1}) \right]$
- For each $i \in [0, \dots, \ell-1]$, For each $j \in [i, \dots, \ell]$, $\hat{W}_i(\beta_j) = W_i(\beta_j) / W_i(\beta_i)$

Given
$\hat{W}_i(\beta_j)$ $\forall i \in \{0, \dots, \ell-1\}$
$\forall j \in \{i, \dots, \ell\}$

We can use
- $\hat{W}_i(x+y) = \hat{W}_i(x) + \hat{W}_i(y)$

to calculate $\hat{W}_i(w_k)$
$\forall i \in \{0, \dots, \ell-1\}$
$\forall k \in \{0, \dots, 2^\ell - 1\}$

⑤ Linear Codes

Defn: A <u>linear code</u> is a code that is a subspace of a vector space over a finite field $\mathbb{F}$

Namely if our code is $C \subseteq \mathbb{F}^n$, and $x, y \in C$, then

$\forall a, b \in \mathbb{F}$, $ax + by \in C$

(setting $a = 1, b = -1$)

$x - y \in C$

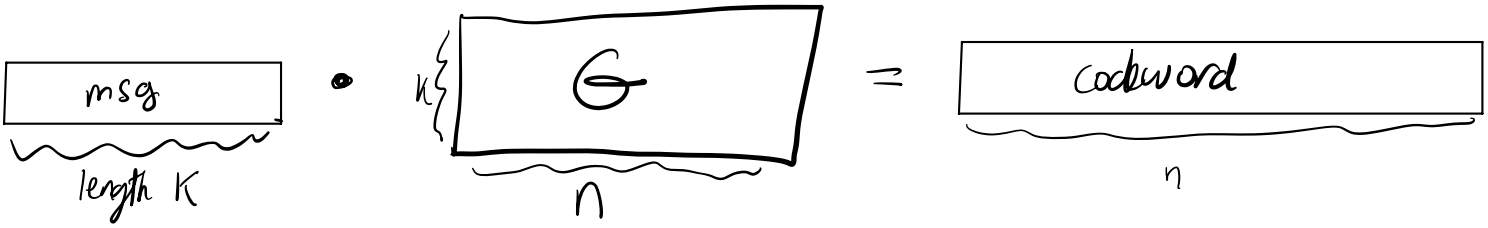
Defn: The Hamming Distance function $\Delta : \mathbb{F}^n \times \mathbb{F}^n \to \{0, \dots, n\}$

$(x, y) \longmapsto$ The number of indices $i \in [0, \dots, n]$ s.t. $x_i \neq y_i$

Defn: The <u>dimension of code C</u> is simply the dimension of C (of the linear subspace)

It is usually denoted by k

Recall if C is a K-dimensional subspace of $\mathbb{F}^n$, then $\exists$ matrix $G \in \mathbb{F}^{K \times n}$



- C is the span of $G_0, \dots, G_{K-1}$, the K rows of G
- a codeword is a linear combination of the rows of G where the K coeffs is the message

Technical note: If you're curious, look up "parity check matrix" H. It's another way to define a linear code.

Defn: Distance of a linear code is $\min_{x,y \in C} \{\Delta(x,y)\}$

or equivalently $\min_{x \in C} \{\Delta(x,0)\}$

Challenge: convince yourself of this equivalence.

Note: a linear code $C \subseteq \mathbb{F}_q^n$ of dimension K, distance d, is denoted as a $[n,K,d]_q$ code

Lemma: Singleton Bound states that every $[n,K,d]_q$ linear code satisfies the inequality

$$d \leq n+1-K$$

when a code achieves equality it's called **MDS**

Defn Maximum Distance Separable (MDS) is when $d = n+1-K$

Defn The rate of a code $\rho := \frac{K}{n} \in (0,1]$

$\rho \uparrow$ redundancy $\downarrow$ space efficiency $\uparrow$

$\rho \downarrow$ redundancy $\uparrow$ space efficiency $\downarrow$

sometimes we care about inv_rate $\in [1, +\infty)$

$= \frac{1}{\rho}$

(6) Reed Solomon Codes:

- Fix some domain $D = \{x_0, \dots, x_{n-1}\} \subseteq \mathbb{F}_q$
- We can define the RS code $[n, K, n+1-K]_q$ like so

univariate!

$$C = \left\{ \left[P(x_0), \dots, P(x_{n-1}) \right] \;\middle|\; P(x) \in \mathbb{F}_q[X]_{<K} \right\}$$

Facts:

- RS codes are linear.

  why? $[P(x_0) + Q(x_0), \dots, P(x_{n-1}) + Q(x_{n-1})]$
  $= [(P+Q)(x_0), \dots, (P+Q)(x_{n-1})]$

  Stop. If this doesn't sit right with you, write out $P(X) = a_0 + a_1 X + a_2 X^2$
  $Q(X) = b_0 + b_1 X + b_2 X^2$
  $R(X) = (a_0+b_0) + (a_1+b_1)X + (a_2+b_2)X^2$
  and convince yourself $\forall x$, $P(x)+Q(x) = R(x)$

- RS codes are MDS, so $d = n+1-K$ always

  why? polynomial magic. Two distinct polynomials of deg $\leq K-1$ can agree on at most $K-1$ places. They disagree at $\geq n-(K-1) = n+1-K$ places.

No reason to write $[n,K,d]_q$, we instead write $RS(n,K)_q$

what happens when $n = K$ (and $d = 1$)?

No, this does not mean that msg is its own codeword

$$m \longmapsto C$$

$m \in \mathbb{F}_q^K$ — encoding — $C \in C \subseteq \mathbb{F}_q^n$

- We are given the msg as some polynomial P coeffs FOR SOME BASIS

let's say basis is $(1, X, X^2, \ldots, X^{K-1})$

- The output is "K evaluations of P over domain D" or... coeffs for P over the "univariate Lagrange basis w.r.t. D"

So basis is $\left(L_{x_0}(x), L_{x_1}(x), \ldots, L_{x_{K-1}}(x)\right)$

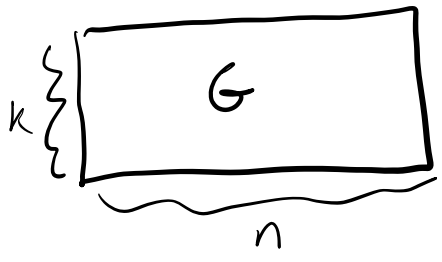
where $L_{x_i}(X) = \begin{cases} 1, & X = x_i \\ 0, & X \in D \setminus \{x_i\} \end{cases}$

"Lagrange polynomial for x_i over D"

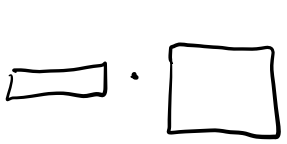
Hmm when $n=K$, the encoding function for RS code is JUST a change of basis..!!

In fact, RS is a linear code, so for specified params $\exists G : \mathbb{F}_q^{K \times n}$ s.t.

$$\begin{aligned} Enc : \mathbb{F}_q^K &\to \mathbb{F}_q^n \\ m &\longmapsto m \cdot G \end{aligned}$$



G is the change of basis matrix.

Typically  takes $O(Kn) = O(n^2)$... but we have a nice 

with great structure embedded.

We can equivalently do $Enc: \mathbb{F}_q^K \to \mathbb{F}_q^n$

$m \longmapsto$ Change of basis to m

and when our bases are chosen well, this will be $O(n \log n)$ 

FOR US, $\mathbb{F}_q = \mathbb{F}_{2^{2^\ell}} = T_\ell$ // binary field

So our msg will be K coeffs w.r.t. Novel Poly Basis
and our codeword will be K coeffs w.r.t. Lagrange Basis Coeffs over $D \subseteq \mathbb{F}_q$, $|D| = K$

What about when $K \neq n$?

We will support K, n as powers of 2 only. $n < 2^{2^\ell}$

Let's say that $n = 2^R \cdot K$ ($R = \log(\frac{1}{\rho})$)

(here R = log_inv_rate)

We will let $D = \{ w_b \mid b \in \{0, \ldots, n-1\} \} \subseteq T_\ell$

Interestingly, NOT only is $D \subseteq T_L$, but it's also a subspace. $D = \langle \beta_0, \ldots, \beta_{\lg(n)-1} \rangle$ ("the span of")

D can be partitioned into 2^R equally sized subsets

$$D_0 = \langle \beta_0, \ldots, \beta_{\lg(K)-1} \rangle = w_0 + D_0$$

$$D_1 = \{ w_K + d \mid d \in D_0 \} = w_K + D_0$$

$$D_2 = \{ w_{2K} + d \mid d \in D_0 \} = w_{2K} + D_0$$

$$\ldots$$

$$D_i = \{ w_{iK} + d \mid d \in D_0 \} = w_{iK} + D_0$$

$$\ldots$$

$$D_{2^R-1} = \{ w_{(2^R-1)K} + d \mid d \in D_0 \} = w_{(2^R-1)K} + D_0$$

called "cosets"

ask Ben/Raju for more!!

$w_K = \beta_{\lg(K)}$

$w_{2K} = \beta_{\lg(K)+1}$

$w_{3K} = \beta_{\lg(K)} + \beta_{\lg(K)+1}$

Note that
- $D = D_0 \cup D_1 \cup \cdots \cup D_{2^R-1}$
- $\forall 0 \le i < j \le 2^R - 1, \; D_i \cap D_j = \emptyset$
- $\forall i, |D_i| = |D_0| = K$

this is what partition means

So, what does our RS over binary field encoding algo look like now?

Input: msg $\vec{m} \in T_\ell^K$, interpreted as K coeffs wrt Novel Poly Basis

Algo:

codeword = [ ]

For coset_shift in $\{0, \ldots, 2^R - 1\}$:

- chunk = AdditiveNTT($\vec{m}$, S = coset_shift * K)
- extend codeword with chunk

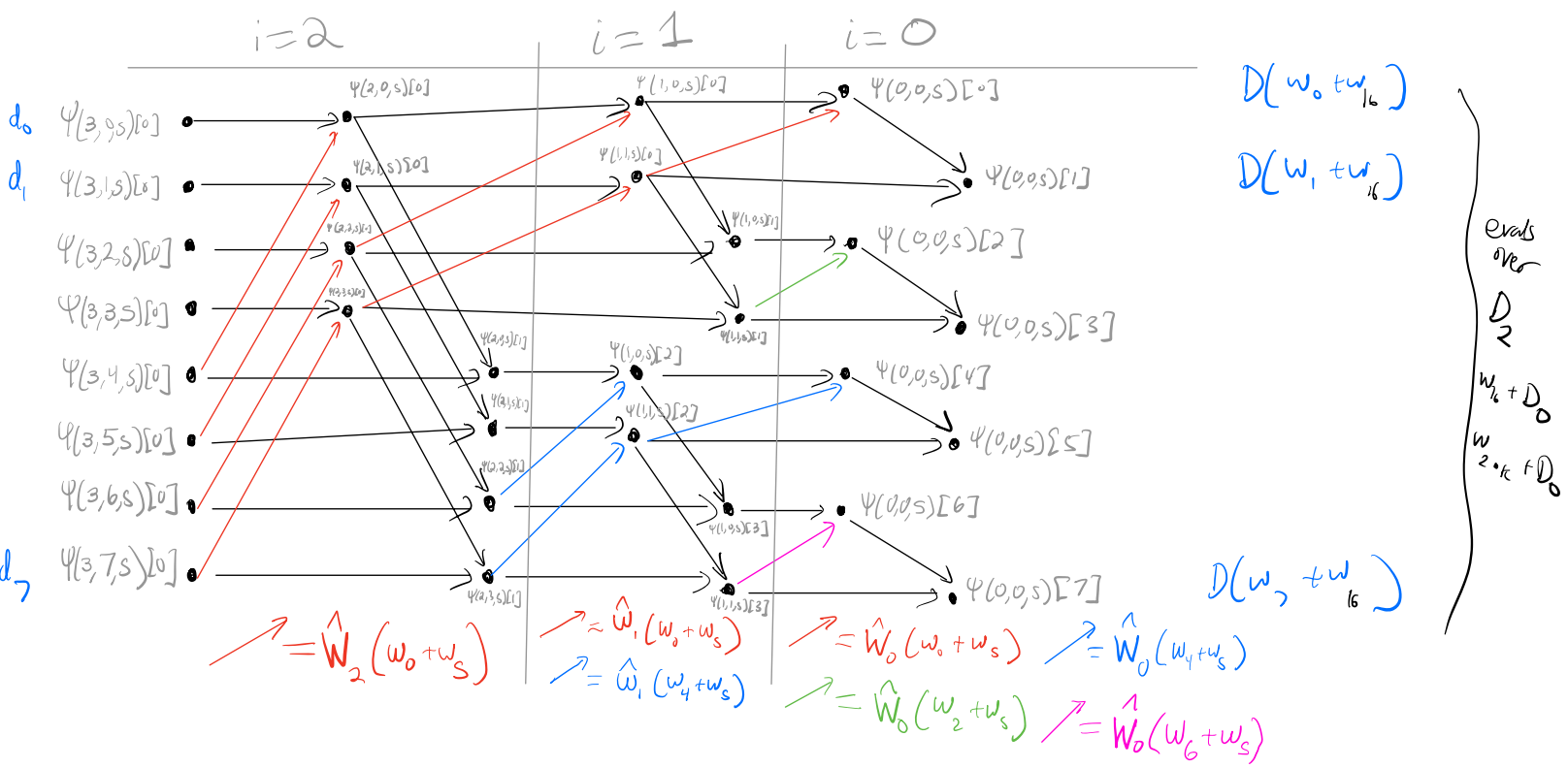
return codeword

Runtime: $\Theta(K \log K \cdot 2^R) = \Theta(n \log K)$

Zooming in to AdditiveNTT(m, s)

ex//

- Forward transform when $\ell = \lg(K)$, $K = 8$, arbitrary S
- $\vec{m} = [\Psi(3,0,s)[0], \ldots \Psi(3,K-1,s)[0]]$



(7) Quick note on inverse transform and decoding

- You can decode from any chunk using iNTT
  Recall the ith chunk of size K is Lagrange Basis coeffs wrt D_i

- The inverse transform follows the exact same recursive structure just starting from

$$c = [\Psi(0,0,s)[0], \ldots, \Psi(0,0,s)[K-1]]$$

$\uparrow$

// $s = K \cdot 2^i$

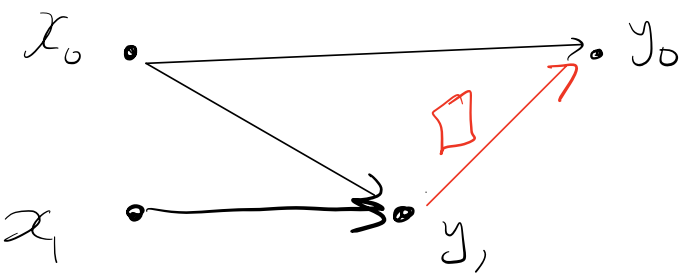
to

$$m = [\Psi(\lg(K), 0, s)[0], \ldots, \Psi(\lg(K), K-1, s)[0]]$$

Appendix:

Additive iNTT

Structure

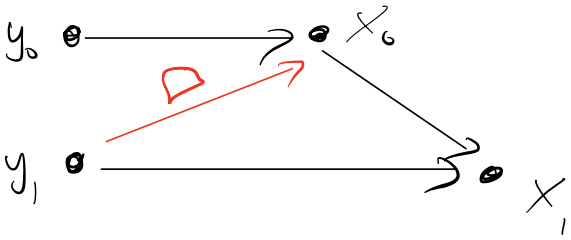


$$y_1 = x_0 + x_1$$

$$y_0 = x_0 + \square y_1$$

Additive NTT

Structure



$$x_0 = y_0 + \square y_1$$

$$x_1 = x_0 + y_1$$

$$(y_0, y_1) \to (y_0 + \square y_1,\ y_0 + \square y_1 + y_1)$$

$$(x_0, x_1) \longmapsto (\square x_0 + \square x_1 + x_0,\ x_0 + x_1)$$

Composition, in characteristic 2, will be identity map