# Decoding Reed–Solomon Codes (of BCH type)

Recall:

**Alphabet:** $\mathcal{A} = \mathbb{F}_q$ for $q$ prime

large $q$ protects against "burst errors"

**Parameters:** $K < N < q$ and generator $\alpha \in \mathbb{F}_q$

* can correct up to $T := N-K$ erasures & up to $\frac{T}{2}$ errors at unknown locations

saturated if $N<q$

* generator polynomial: $G = (X-\alpha)\cdots(X-\alpha^T)$

**Encoder:** Input: $s^K \in \mathcal{A}^K$

* $P \leftarrow s_1 + s_2 X + \dots + s_K X^{K-1}$
* $R \leftarrow P \cdot X^T \bmod G$ — remainder of poly division (degree $< T$)
* $M \leftarrow P \cdot X^T - R$ — degree $N-1$
* $x^N \leftarrow$ coefficients of $M$ — i.e. $M = x_1 + x_2 X + \dots + x_N X^{N-1}$

By construction:

* $x^N = [x_1, .., x_T, s_1, .., s_K]$ — $M$ and $P \cdot X^T$ differ in degree $< T$ only!
* $M$ is multiple of $G \Longrightarrow \boxed{M(\alpha) = \dots = M(\alpha^T) = 0}$ (⊛)

ex: $K=1$, $N=3$, $q=5$ and $\alpha = 2$

↳ $T=2$ & $G = (X-2)(X+1) = X^2 - X - 2$ ← $-2 \equiv 3 \pmod 5$ etc

↳ $s \in \mathbb{F}_5$ is encoded into $x^N(s) = [-2s, -s, s]$

## How to decode?

Imagine we receive $y^N \in \mathcal{A}^N$. Interpret as coeffs of polynomial:

$$R = M + E$$

with error polynomial

$$E = \sum_{k=1}^{\mathcal{C}} e_k X^{i_k}$$

$\mathcal{C}$: #errors; $i_k$: locations $\in \{0, \dots, N-1\}$ grr...; $e_k$: mismatch

## Two settings:

* Erasures: $e_k$ unknown, $C$ and $i_k$ known
* General errors: everything unknown

What do we know? ⊛ implies:

$$\text{(I)}\quad \left\{ \begin{aligned} E(\alpha) &= \sum_{k=1}^{C} e_k \alpha^{i_k} = R(\alpha) \\ &\;\vdots \\ E(\alpha^T) &= \sum_{k=1}^{C} e_k \alpha^{T\cdot i_k} = R(\alpha^T) \end{aligned} \right\}$$

$T$ linear equations in $e_1, \dots, e_C$ ... if locations $i_1, \dots, i_C$ known

This solves the problem for erasure errors: Can correct $\boxed{C \le T \text{ erasures}}$

**ex:** $x^N = [\cancel{-2s}, -s, \cancel{s}]$

assume $T=2$ erasure errors:

known locations

$\rightsquigarrow y^N = [0, -s, 0] \rightsquigarrow R = -sX \qquad E = e_1 X^0 + e_2 X^2 = e_1 + e_2 X^2$

$E(2) = e_1 + e_2 4 \overset{!}{=} R(2) = -2s$

$E(4) = e_1 + e_2 \overset{!}{=} R(4) = s \quad \Rightarrow \quad e_1 = 2s, \; e_2 = -s, \; E = 2s - sX^2$

$\Rightarrow M = R - E = -2s - sX + sX^2 \mathrel{\hat{=}} [-2s, -s, s]$ ☺

(But e.g. $x^N = [\cancel{x}, 0, \cancel{x}] \Rightarrow$ impossible to correct ↯)

**Decoder for erasures:** Input: $y^N \in \mathcal{A}^N$, error locations $i_1, \dots, i_C$

* $R \leftarrow y_1 + y_2 Z + \dots + y_N Z^{N-1}$
* Solve (I) for $e_1, \dots, e_C$
* $E \leftarrow e_1 X^{i_1} + \dots + e_C X^{i_C}$
* $M \leftarrow R - E$
* $\hat{S}^K \leftarrow$ leading $K$ coeffs of $M$ (i.e. $\hat{S}_1 = m_{N-K+1}, \dots, \hat{S}_K = m_N$)

$e_1 - e_2 = -2s$

$e_1 + e_2 = s$

$2e_1 = -s \qquad e_1 = -3s = 2s$

What if locations unknown? Consider locator polynomial:

Should all be distinct: need $N \le q-1$

$$L := \prod_{k=1}^{c_i} \left(1 - X\alpha^{i_k}\right) = 1 + L_1 X + \dots + L_c X^c$$

Roots are $\alpha^{-i_k}$ for $k = 1, \dots, c_i$. How to determine $L$?

$$0 = \sum_k e_k \alpha^{i_k(j+c_i)} L(\alpha^{-i_k})$$

$$= E(\alpha^{j+c_i}) + L_1 \cdot E(\alpha^{j+c_i-1}) + \dots + L_c \cdot E(\alpha^{j})$$

But: $E(\alpha) = R(\alpha), \dots, E(\alpha^T) = R(\alpha^T)$:

$$(2) \quad \begin{bmatrix} R(\alpha^c) & \cdots & R(\alpha) \\ \vdots & & \vdots \\ R(\alpha^{2c-1}) & \cdots & R(\alpha^c) \end{bmatrix} \begin{bmatrix} L_1 \\ \vdots \\ L_c \end{bmatrix} = \begin{bmatrix} -R(\alpha^{c+1}) \\ \\ -R(\alpha^{2c_i}) \end{bmatrix}$$

← linear system for $L_1, \dots, L_c$

... as long as $2c_i \le T$, i.e., $\boxed{c_i \le \frac{T}{2} \text{ errors}}$.

Still don't know $c_i$ — so just try from $c_i = \lfloor \frac{T}{2} \rfloor, \dots, 1$ until (2) unique solution.

Once we know $L$: search roots $\alpha^{-i_k} \rightsquigarrow i_k \rightsquigarrow e_k \rightsquigarrow E$.

$\boxed{\text{ex:}}$ $S = 1$ is encoded in $x^N = [-2, -1, 1]$

Assume we receive $y^N = [-2, -1, 0] \rightsquigarrow R = -2 - X$

$R(\alpha) = 1 \ne 0$

$R(\alpha^2) = -1 \ne 0$

↳ error(s) happened.

Try $\boxed{c_i = 1:}$

(2): $\overset{1}{R(\alpha)} \cdot L_1 = -\overset{-1}{R(\alpha^2)}$

$\Longrightarrow L_1 = 1$, i.e. $L = 1 + X$

→ $L$ has root $g_1 = 4 = \alpha^2 = \alpha^{-2}$

↳ location $i_1 = 2$ → $E = eX^2$

(1): $E(\alpha) = 1$ $\Longrightarrow$ $e = -1$, $E = -X^2$

$E(\alpha^2) = -1$

$\Longrightarrow M = R - E = -2 - X + X^2 \,\hat{=}\, [-2, -1, 1]$

Result:

$\boxed{\text{Decoder:}}$ Input: $y^N \in \mathcal{A}^N$

* $R \leftarrow y_1 + y_2 z + \dots + y_N z^{N-1}$
* If $R(\alpha) = \dots = R(\alpha^T) = 0$: $M \leftarrow R$

  else:

  For $C = \lfloor \frac{T}{2} \rfloor, \dots, 1$:

  If Det $= 0$ in (2): continue

  Solve (2) for $L_1, \dots, L_C$

  $L \leftarrow 1 + L_1 z + \dots + L_C z^C$

  $\beta_1, \dots, \beta_C \leftarrow$ roots of $L$ ← — search

  For $k = 1, \dots, C$:

  $i_k \leftarrow$ number in $\{0, \dots, N-1\}$ s.th. $\beta_k = \alpha^{-i_k}$ $= \alpha^{q-1-i_k}$ ← — search/look up

  Solve (1) for $e_1, \dots, e_C$

  $E \leftarrow \sum_{k=1}^{C} e_k z^{i_k}$

  $M \leftarrow R - E$

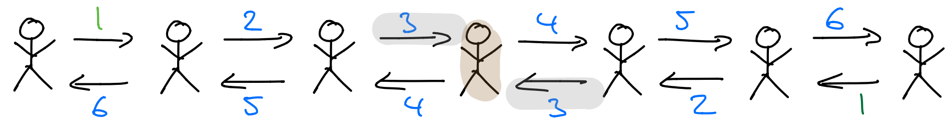
  Break

* $\hat{S}^K \leftarrow$ leading $K$ coeffs of $M$ (i.e. $\hat{S}_1 = m_{N-K+1}, \dots, \hat{S}_K = m_N$)

# Message Passing Algorithms (§16)    Cf. dynamic programming

Motivation: Iterative algos to compute/approximate/maximize $P(s|y^N)$!

**Decentralized counting:**

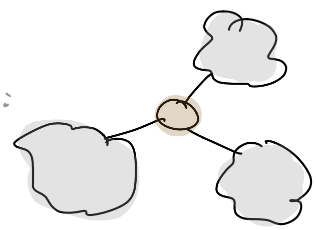



$\boxed{\begin{array}{l}\text{if } 1^{st}\text{: send } 1 \text{ to back} \\ \text{if last: send } 1 \text{ to front} \\ \text{if receive message } m\text{: send } m+1 \text{ to other neighbor} \\ \text{if received messages from all neighbors: output } \sum +1\end{array}}$

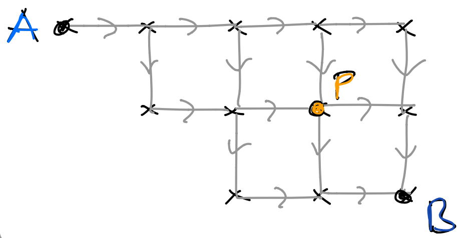
every node runs this algo

Separability property: total = #left + #right + 1

This extends easily to trees (= graphs w/o cycles):



**Paths in a grid:** Consider paths from A to B with each step → or ↓




#=9

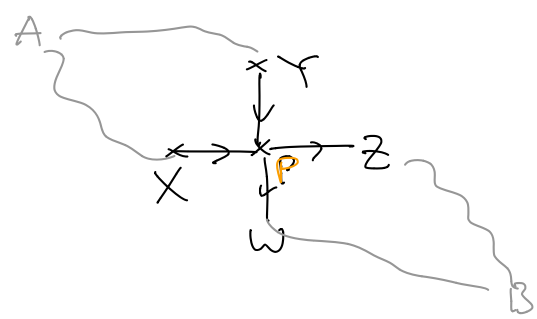
Objectives:

① Count # paths from A → B!

② Count #paths A → P → B!

③ Sample path A → B uniformly at random!

① Separation properties:




$\#\{A \to P\} = \#\{A \to X\} + \#\{A \to Y\}$

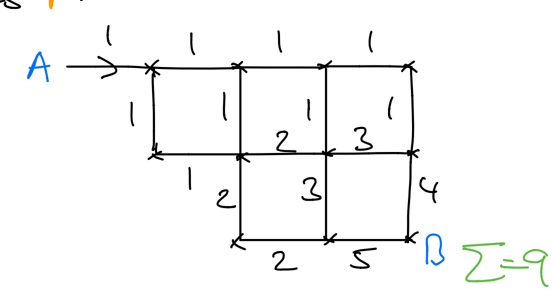
$\#\{P \to B\} = \#\{Z \to B\} + \#\{W \to B\}$

Two options:

**Forward algo:** Node A sends 1. All other nodes P:

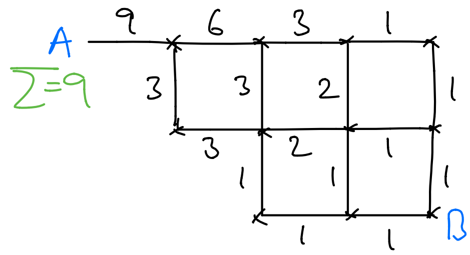
$\boxed{\begin{array}{l}\text{Wait until message from all upstream received} \\ \text{Send } \sum \text{ downstream}\end{array}}$

$\sum$ ← $\#\{A \to P\}$



**Backward algo:** Node B sends 1. All other nodes P:

$$\boxed{\text{Wait until message from all downstream received; Send } \Sigma \text{ upstream}}$$




$\uparrow \#\{A \to P\}$

② Separation property: $\#\{A \to P \to B\} = \#\{A \to P\} \cdot \#\{P \to B\}$

from above

↳ Can compute

$$\boxed{\Pr(\text{path through } P) = \frac{\#\{A \to P \to B\}}{\#\{A \to B\}}}$$

after forward AND backward pass.

③ Run backward pass. Then, sample node by node: $P_0 \equiv A$

$$P(P_{\ell+1} \mid P_\ell) = \frac{\#\{P_{\ell+1} \to B\}}{\#\{P_\ell \to B\}}$$

where $P_{\ell+1}$ downstream neighbor of $P_\ell$