Coduri Reed-Solomon

Buzatu Giulian - 352
Ilie Dumitru - 352
Preda Maria - 351

31 ianuarie 2025

Rezumat

În această lucrare vom prezenta codurile Reed-Solomon, o metodă de codificare a datelor care permite detectarea și corectarea erorilor apărute în cadrul transmiterii unui mesaj. Vom descrie modul de codificare a unui mesaj și două moduri de decodificare: varianta brută și algoritmul Berlekamp-Welch. Documentul include exemple și explicații detaliate pentru o înțelegere clară a acestor procedee.

Cuprins

1 Introducere

Codurile Reed-Solomon au apărut pentru prima dată în anul 1960, fiind introduse de Irving S. Reed și Gustave Solomon [Wik25] . Acestea reprezintă un procedeu de codificare a datelor (datelor in loc de mesajelor), care permite detectarea și corectarea erorilor care apar în transmiterea mesajelor.

Dacă dorim să transmitem un mesaj de lungime k și să putem corecta cel mult s erori apărute în cadrul transmiterii, atunci vom trimite un mesaj de dimensiune $k + 2s$.

Proprietatea codurilor Reed-Solomon de a corecta erorile este utilizată în aplicații (aplicații în loc de domenii) precum CD-uri, DVD-uri și QR codes. Spre exemplu, un CD se poate deteriora, dar sunt cazuri când acesta încă poate fi folosit. Acest lucru se întâmplă cu ajutorul codurilor Reed-Solomon, prin intermediul cărora se poate găsi și corecta informația greșită. Totuși, după un anumit prag de deteriorare, avem prea multă informație greșită care nu se mai poate recupera.

2 Codificare

Există mai multe variante de codificare când vine vorba de coduri Reed-Solomon, noi am ales un mod de codificare care se bazează pe trimiterea valorilor unui polinom în anumite puncte $x_0, x_1, x_2, \cdots, x_{k+2s-1}$ stabilite de comun acord între cel care codifică și cel care decodifică. În continuare, vom considera aceste puncte ca fiind $0, 1, 2, \cdots, k + 2s - 1$.

Să presupunem că vrem să stocăm date pe un mediu ce poate avea probleme de pierdere sau modificări accidentale. Pentru a preveni pierderea datelor vom introduce redundanță în modelul de stocare.

Ne va fi mai ușor să discutăm despre aceste date ca un șir de octeți sau, mai comun, simboluri.

2.1 Algoritmul de codificare

Fie șirul de simboluri $y_0, y_1, \cdots, y_{k-1}$ pe care vrem să-l codificăm. Vom vrea să găsim un polinom special P de grad $k - 1$, astfel încât

$$P(0) = y_0, P(1) = y_1, \cdots, P(k-1) = y_{k-1}$$

Pentru a adăuga redundanța și a permite corectarea a cel mult s erori, vom calcula valorile

$$P(k) = y_k, P(k+1) = y_{k+1}, \cdots, P(k+2s-1) = y_{k+2s-1}$$

În final, vom stoca valorile

$$y_0, y_1, \cdots, y_{k-1}, P(k), P(k+1), \cdots, P(k+2s-1)$$

Datorită introducerii redundanței, acum putem, bazat pe oricare k simboluri trimise, să îl găsim pe P, deci putem găsi mesajul inițial, anume, $y_0, y_1, \cdots, y_{k-1}$.[Ver22]

2.2 Calcularea polinomului P

Date fiind $y_0, y_1, \cdots, y_{k-1}$, care reprezintă valorile polinomului P în punctele $0, 1, \cdots, k-1$ există două metode de găsire a polinomului P:

1. Eliminare gaussiană
2. Interpolare lagrangiană

2.2.1 Eliminare gaussiană

Ne dorim să rezolvăm următoarul sistem de ecuații:

$$\begin{cases} P(0) = y_0 \\ P(1) = y_1 \\ \cdots \\ P(k-1) = y_{k-1} \end{cases} \iff \begin{cases} p_0 + p_1 \cdot 0 + p_2 \cdot 0^2 + \cdots + p_{k-1} \cdot 0^{k-1} = y_0 \\ p_0 + p_1 \cdot 1 + p_2 \cdot 1^2 + \cdots + p_{k-1} \cdot 1^{k-1} = y_1 \\ \cdots \\ p_0 + p_1 \cdot (k-1) + p_2 \cdot (k-1)^2 + \cdots + p_{k-1} \cdot (k-1)^{k-1} = y_{k-1} \end{cases}.$$

Putem scrie acest sistem de ecuații și sub formă matriceală:

$$A \cdot p = Y \iff \begin{pmatrix} 1 & 0^1 & \cdots & 0^{k-1} \\ 1 & 1^1 & \cdots & 1^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & (k-1)^1 & \cdots & (k-1)^{k-1} \end{pmatrix} \cdot \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_{k-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{k-1} \end{pmatrix}$$

Ne dorim să scriem fiecare necunoscută doar în funcție de necunoscutele cu indice mai mare. Prin urmare, vom folosi aplicații elementare pentru a transforma matricea extinsă astfel încât să avem doar valori de 0 sub diagonala formată din $a_{0,0}, a_{1,1}, \cdots, a_{k-1,k-1}$:

$$\begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & \cdots & a_{0,k-1} & y_0 \\ a_{1,0} & a_{1,1} & a_{1,2} & \cdots & a_{1,k-1} & y_1 \\ a_{2,0} & a_{2,1} & a_{2,2} & \cdots & a_{2,k-1} & y_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{k-1,1} & a_{k-1,2} & a_{k-1,3} & \cdots & a_{k-1,k-1} & y_{k-1} \end{pmatrix} \Longrightarrow \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & \cdots & a_{0,k-1} & y_0 \\ 0 & a'_{1,1} & a'_{1,2} & \cdots & a'_{1,k-1} & y'_1 \\ 0 & 0 & a'_{2,2} & \cdots & a'_{2,k-1} & y'_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a'_{k-1,k-1} & y'_{k-1} \end{pmatrix}$$

Acum, vom afla $p_{k-1}, p_{k-2}, \cdots, p_0$, în această ordine, folosind următoarea formulă:

$$p_i = \frac{y'_i - \sum_{j=i+1}^{n} a'_{i,j} \cdot p_j}{a'_{i,i}}.$$

Complexitatea algoritmului este $O(k^3)$.[Tr3]

2.2.2 Interpolare lagrangiană

Pentru a găsi polinomul P, ne dorim ca pentru fiecare $i \in \{0, 1, \cdots, k-1\}$ să avem un polinom care să aibă valoarea y_i pentru inputul i și 0 pentru inputurile $0, 1, \cdots i-1, i+1, \cdots, k-1$.

Pentru a construi polinomul dorit pentru $i = 0$, vom construi mai întâi un polinom care să aibă valoarea 0 în punctele $1, 2, \cdots, k-1$. Acest lucru este ușor, polinomul dorit este doar produsul de monoame

$$Q_0(x) = (x-1)(x-2)\cdots(x-(k-1)).$$

Acum, putem construi polinomul dorit astfel:

$$R_0(x) = \frac{Q_0(x) \cdot y_0}{Q_0(0)}.$$

Se poate observa că $R_0(0) = y_0$ și $R_0(j) = 0, \forall j \in \{1, 2, \cdots, k-1\}$.
Analog, construim Q_i și $R_i, \forall i \in \{1, 2, \cdots, k-1\}$.

$$Q_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{k-1} (x-j)$$

$$R_i(x) = \frac{Q_i(x) \cdot y_i}{Q_i(i)}$$

Astfel, polinomul P va fi construit în felul următor:

$$P = \sum_{i=0}^{k-1} R_i(x).$$

Complexitatea algoritmului implementat cum am descris mai sus este $O(k^3)$. Complexitatea ridicată se datorează înmulțirilor de polinoame. Dacă pentru calcularea polinoamelor Q_i precalculăm produsul $Q = \sum_{j=0}^{k-1} (x-j)$ și apoi doar facem împărțirea polinomului la monomul $(x-i)$ în mod deștept, putem obține complexitatea $O(k^2)$. [Wik24]

2.2.3 Eliminare gaussiană vs. Interpolare lagrangiană

Preferăm să folosim interpolare lagrangriană din mai multe motive:

- este o diferență considerabilă între $O(k^2)$ și $O(k^3)$
- folosim mai puțină memorie, deoarece nu e nevoie să reținem o matrice de dimensiune (k, k)

2.3 De ce folosim un polinom pentru codificare?

Principalele motive pentru care folosim un polinom când vine vorba de codificare sunt proprietățile algebrice ale acestuia.

Este important că putem codifica mesaje oricât de mari și să adăugăm oricât de multe valori adiționale ne dorim pentru a permite corecția erorilor, iar acest lucru este posibil datorită modului ușor de adăugare a unei noi valori, anume calcularea valorii polinomului pentru un anumit input.

De asemenea, folosirea unui polinom nu aduce nicio condiție suplimentară referitoare la locul în care o eroare se poate produce în mesaj, ci contează doar numărul lor.

2.4 Exemplu

Dorim să codificăm mesajul $Y = (7, 8, 10, 2)$ de lungime $k = 4$ și să permitem corectarea a $s = 1$ erori. Vom prezenta cele două moduri de a determina polinomul P. Întâi folosind eliminare gaussiană. Avem următorul calcul matriceal:

$$\begin{pmatrix} 1 & 0^1 & 0^2 & 0^3 \\ 1 & 1^1 & 1^2 & 1^3 \\ 1 & 2^1 & 2^2 & 2^3 \\ 1 & 3^1 & 3^2 & 3^3 \end{pmatrix} \cdot \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_{k-1} \end{pmatrix} = \begin{pmatrix} 7 \\ 8 \\ 10 \\ 2 \end{pmatrix}$$

Vom aduce matricea extinsă în forma dorită astfel:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 7 \\ 1 & 1 & 1 & 1 & 8 \\ 1 & 2 & 4 & 8 & 10 \\ 1 & 3 & 9 & 27 & 2 \end{pmatrix} \Longrightarrow \begin{pmatrix} 1 & 0 & 0 & 0 & 7 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 2 & 4 & 8 & 3 \\ 0 & 3 & 9 & 27 & -5 \end{pmatrix} \tag{1}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 7 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 2 & 4 & 8 & 3 \\ 0 & 3 & 9 & 27 & -5 \end{pmatrix} \Longrightarrow \begin{pmatrix} 1 & 0 & 0 & 0 & 7 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 2 & 6 & 1 \\ 0 & 0 & 6 & 24 & -8 \end{pmatrix} \tag{2}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 7 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 2 & 6 & 1 \\ 0 & 0 & 6 & 24 & -8 \end{pmatrix} \Longrightarrow \begin{pmatrix} 1 & 0 & 0 & 0 & 7 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 2 & 6 & 1 \\ 0 & 0 & 0 & 6 & -11 \end{pmatrix} \tag{3}$$

În cadrul ecuației (1), ne dorim să avem 0 pe prima coloană pe toate liniile de sub linia 0, prin urmare, liniile matricei s-au schimbat astfel:

$$L_1 = L_1 - L_0$$
$$L_2 = L_2 - L_0$$
$$L_3 = L_3 - L_0$$

În cadrul ecuației (2), ne dorim să avem 0 pe a doua coloană pe toate liniile de sub linia 1, prin urmare, liniile matricei s-au schimbat astfel:

$$L_2 = L_2 - 2 \cdot L_1$$
$$L_3 = L_3 - 3 \cdot L_1$$

În cadrul ecuației (3), ne dorim să avem 0 pe a treia coloană pe toate liniile de sub linia 2, prin urmare, liniile matricei s-au schimbat astfel:

$$L_3 = L_3 - 3 \cdot L_2$$

Acum că avem matricea în forma dorită, putem afla coeficienții polinomului:

$$p_3 = -\frac{11}{6}$$
$$p_2 = \frac{1 - 6 \cdot p_3}{2} = \frac{12}{2} = 6$$
$$p_1 = \frac{1 - (1 \cdot p_2 + 1 \cdot p_3)}{1} = 1 - (6 - \frac{11}{6}) = 1 - \frac{25}{6} = -\frac{19}{6}$$
$$p_0 = \frac{7 - (0 \cdot p_1 + 0 \cdot p_2 + 0 \cdot p_3)}{1} = 7$$

Putem observa că în urma transformării matricei inițiale, $p_0 = y_0$, deoarece pe linia 0 toți termenii, cu excepția lui $a_{0,0}$, sunt 0.

În urma calculelor, am obținut polinomul $P(x) = -\frac{11}{6}x^3 + 6x^2 - \frac{19}{6}x + 7$.

În continuare prezentăm aflarea polinomului P folosind interpolarea lagrangiană. Definim polinoamele Q_0, Q_1, Q_2 și Q_3 astfel:

$$Q_0(x) = (x-1)(x-2)(x-3)$$
$$Q_1(x) = (x-0)(x-2)(x-3)$$
$$Q_2(x) = (x-0)(x-1)(x-3)$$
$$Q_3(x) = (x-0)(x-1)(x-2)$$

Folosind cele 4 polinoame Q_i, definim polinoamele R_0, R_1, R_2 și R_3:

$$R_0(x) = \frac{Q_0(x) \cdot y_0}{Q_0(0)} = \frac{(x-1)(x-2)(x-3) \cdot 7}{(-1) \cdot (-2) \cdot (-3)} = -\frac{7(x-1)(x-2)(x-3)}{6}$$
$$= -\frac{7}{6}x^3 + 7x^2 - \frac{77}{6}x + 7$$

$$R_1(x) = \frac{Q_1(x) \cdot y_1}{Q_1(1)} = \frac{(x-0)(x-2)(x-3) \cdot 8}{1 \cdot (-1) \cdot (-2)} = \frac{8x(x-2)(x-3)}{2} = 4x(x-2)(x-3)$$
$$= 4x^3 - 20^2 + 24x$$

$$R_2(x) = \frac{Q_2(x) \cdot y_2}{Q_2(2)} = \frac{(x-0)(x-1)(x-3) \cdot 10}{2 \cdot 1 \cdot (-1)} = -\frac{10x(x-1)(x-3)}{2} = -5x(x-1)(x-3)$$
$$= -5x^3 + 20x^2 - 15x$$

$$R_3(x) = \frac{Q_3(x) \cdot y_3}{Q_3(3)} = \frac{(x-0)(x-1)(x-2) \cdot 2}{3 \cdot 2 \cdot 1} = \frac{2x(x-1)(x-2)}{6} = \frac{x(x-1)(x-2)}{3}$$
$$= \frac{1}{3}x^3 - x^2 + \frac{2}{3}x$$

Acum putem afla polinomul P astfel:

$$
\begin{aligned}
P &= \sum_{i=0}^{3} R_i(x) \\
&= -\frac{7}{6}x^3 + 7x^2 - \frac{77}{6}x + 7 + 4x^3 - 20x^2 + 24x - 5x^3 + 20x^2 - 15x + \frac{1}{3}x^3 - x^2 + \frac{2}{3}x \\
&= (-\frac{7}{6} + 4 - 5 + \frac{1}{3})x^3 + (7 - 20 + 20 - 1)x^2 + (-\frac{77}{6} + 24 - 15 + \frac{2}{3})x + 7 \\
&= -\frac{11}{6}x^3 + 6x^2 - \frac{19}{6}x + 7
\end{aligned}
$$

Observăm că ambele metode au găsit același polinom. Acest lucru se întâmplă deoarece polinomul rezultat este **unic**. Pentru a permite corectarea a $s = 1$ erori, trebuie să calculăm încă două valori ale polinomului, anume $P(4)$ și $P(5)$.

$$
\begin{aligned}
P(4) &= -\frac{11}{6} \cdot 64 + 6 \cdot 16 - \frac{19}{6} \cdot 4 + 7 = \cdots = -27 \\
P(5) &= -\frac{11}{6} \cdot 125 + 6 \cdot 25 - \frac{19}{6} \cdot 5 + 7 = \cdots = -88
\end{aligned}
$$

Așadar, vom transmite următorul mesaj $Y' = (7, 8, 10, 2, -27, -88)$.

3 Decodificare

Există mai multe moduri de a decodifica datele și a obține mesajul inițial $y_0, y_1, \cdots, y_{k-1}$. În continuare, vom prezenta două moduri diferite de decodificare a mesajului.

3.1 Varianta brut

Nu putem spune exact care simboluri au fost modificate, însă putem să facem mai multe încercări. Pașii algoritmului sunt următorii:

- Alegem k elemente din setul de date.
- Generăm polinomul Q de grad $k - 1$ care ar genera aceste puncte (folosind una dintre metodele prezentate la codificare).
- Polinomul Q primește un vot.
- Repetăm pentru toate celelalte moduri de alegere a k elemente.

La final cel mai votat polinom este cel considerat corect. În cazul în care apar mai mult de s erori, polinomul rezultat nu va fi cel folosit pentru codificare.

Iterarea prin toate modurile de a alege k elemente din $k + 2s$ durează $\binom{k+2s}{k} = \frac{(k+2s)!}{k!(2s)!}$. Calcularea polinomului pentru un set de k elemente durează $O(k^2)$.[Ver22]

3.1.1 Exemple

Vom considera mesajul codificat mai devreme, primit o dată fără erori, o dată cu numărul maxim de erori și o dată cu un număr prea mare de erori.

- Exemplul 1

  Am primit mesajul $(7, 8, 10, 2, -27, -88)$ (observăm că este același cu cel codificat), cu $k = 4$ și $s = 1$.

  Toate posibilitățile de alegere a 4 perechi de forma (x, y) sunt:

  - $(0, 7), (1, 8), (2, 10), (3, 2) \implies Q = -\frac{11}{6}x^3 + 6x^2 - \frac{19}{6}x + 7$

- $(0,7),(1,8),(2,10),(4,-27) \implies Q = -\frac{11}{6}x^3 + 6x^2 - \frac{19}{6}x + 7$
- $(0,7),(1,8),(2,10),(5,-88) \implies Q = -\frac{11}{6}x^3 + 6x^2 - \frac{19}{6}x + 7$
- $(0,7),(1,8),(3,2),(4,-27) \implies Q = -\frac{11}{6}x^3 + 6x^2 - \frac{19}{6}x + 7$
- $(0,7),(1,8),(3,2),(5,-88) \implies Q = -\frac{11}{6}x^3 + 6x^2 - \frac{19}{6}x + 7$
- $(0,7),(1,8),(4,-27),(5,-88) \implies Q = -\frac{11}{6}x^3 + 6x^2 - \frac{19}{6}x + 7$
- $(0,7),(2,10),(3,2),(4,-27) \implies Q = -\frac{11}{6}x^3 + 6x^2 - \frac{19}{6}x + 7$
- $(0,7),(2,10),(3,2),(5,-88) \implies Q = -\frac{11}{6}x^3 + 6x^2 - \frac{19}{6}x + 7$
- $(0,7),(2,10),(4,-27),(5,-88) \implies Q = -\frac{11}{6}x^3 + 6x^2 - \frac{19}{6}x + 7$
- $(0,7),(3,2),(4,-27),(5,-88) \implies Q = -\frac{11}{6}x^3 + 6x^2 - \frac{19}{6}x + 7$
- $(1,8),(2,10),(3,2),(4,-27) \implies Q = -\frac{11}{6}x^3 + 6x^2 - \frac{19}{6}x + 7$
- $(1,8),(2,10),(3,2),(5,-88) \implies Q = -\frac{11}{6}x^3 + 6x^2 - \frac{19}{6}x + 7$
- $(1,8),(2,10),(4,-27),(5,-88) \implies Q = -\frac{11}{6}x^3 + 6x^2 - \frac{19}{6}x + 7$
- $(1,8),(3,2),(4,-27),(5,-88) \implies Q = -\frac{11}{6}x^3 + 6x^2 - \frac{19}{6}x + 7$
- $(2,10),(3,2),(4,-27),(5,-88) \implies Q = -\frac{11}{6}x^3 + 6x^2 - \frac{19}{6}x + 7$

Observăm că de fiecare dată am obținut același polinom, care este și polinomul cu care am codificat mesajul, deci acesta are număr maxim de voturi și este cel ales pentru decodificare.

- Exemplul 2

  Am primit mesajul $(7,8,10,2,-32,-88)$ (observăm că valoarea y_4 este diferită), cu $k = 4$ și $s = 1$.

  Toate posibilitățile de alegere a 4 perechi de forma (x, y) sunt:

  - $(0,7),(1,8),(2,10),(3,2) \implies Q = -\frac{11}{6}x^3 + 6x^2 - \frac{19}{6}x + 7$
  - $(0,7),(1,8),(2,10),(4,-32) \implies Q = -\frac{49}{24}x^3 + \frac{53}{8}x^2 - \frac{43}{12}x + 7$
  - $(0,7),(1,8),(2,10),(5,-88) \implies Q = -\frac{11}{6}x^3 + 6x^2 - \frac{19}{6}x + 7$
  - $(0,7),(1,8),(3,2),(4,-32) \implies Q = -\frac{9}{4}x^3 + \frac{23}{3}x^2 - \frac{53}{12}x + 7$
  - $(0,7),(1,8),(3,2),(5,-88) \implies Q = -\frac{11}{6}x^3 + 6x^2 - \frac{19}{6}x + 7$
  - $(0,7),(1,8),(4,-32),(5,-88) \implies Q = -\frac{17}{12}x^3 + \frac{7}{2}x^2 - \frac{13}{12}x + 7$
  - $(0,7),(2,10),(3,2),(4,-32) \implies Q = -\frac{59}{24}x^3 + \frac{73}{8}x^2 - \frac{83}{12}x + 7$
  - $(0,7),(2,10),(3,2),(5,-88) \implies Q = -\frac{11}{6}x^3 + 6x^2 - \frac{19}{6}x + 7$
  - $(0,7),(2,10),(4,-32),(5,-88) \implies Q = -\frac{29}{24}x^3 + \frac{13}{8}x^2 + \frac{37}{12}x + 7$
  - $(0,7),(3,2),(4,-32),(5,-88) \implies Q = -\frac{7}{12}x^3 - 4x^2 + \frac{187}{12}x + 7$
  - $(1,8),(2,10),(3,2),(4,-32) \implies Q = -\frac{8}{3}x^3 + 11x^2 - \frac{37}{3}x + 12$
  - $(1,8),(2,10),(3,2),(5,-88) \implies Q = -\frac{11}{6}x^3 + 6x^2 - \frac{19}{6}x + 7$
  - $(1,8),(2,10),(4,-32),(5,-88) \implies Q = -x^3 - \frac{2}{3}x^2 + 11x - \frac{4}{3}$
  - $(1,8),(3,2),(4,-32),(5,-88) \implies Q = -\frac{1}{6}x^3 - 9x^2 + \frac{211}{6}x - 18$
  - $(2,10),(3,2),(4,-32),(5,-88) \implies Q = \frac{2}{3}x^3 - 19x^2 + \frac{223}{3}x - 68$

  Observăm că polinomul cu cele mai multe voturi este $-\frac{11}{6}x^3+6x^2-\frac{19}{6}x+7$, același ca la codificare, și prin urmare va fi folosit pentru decodificare.

- Exemplul 3

  Am primit mesajul $(7,8,30,2,-32,-88)$ (observăm că valorile y_2 și y_4 sunt diferite), cu $k = 4$ și $s = 1$.

  Toate posibilitățile de alegere a k perechi de forma (x, y) sunt:

- $(0,7),(1,8),(2,30),(3,2) \implies Q = -\frac{71}{6}x^3 + 46x^2 - \frac{199}{6}x + 7$
- $(0,7),(1,8),(2,30),(4,-32) \implies Q = -\frac{169}{24}x^3 + \frac{253}{8}x^2 - \frac{283}{12}x + 7$
- $(0,7),(1,8),(2,30),(5,-88) \implies Q = -\frac{31}{6}x^3 + 26x^2 - \frac{119}{6}x + 7$
- $(0,7),(1,8),(3,2),(4,-32) \implies Q = -\frac{9}{4}x^3 + \frac{23}{3}x^2 - \frac{53}{12}x + 7$
- $(0,7),(1,8),(3,2),(5,-88) \implies Q = -\frac{11}{6}x^3 + 6x^2 - \frac{19}{6}x + 7$
- $(0,7),(1,8),(4,-32),(5,-88) \implies Q = -\frac{17}{12}x^3 + \frac{7}{2}x^2 - \frac{13}{12}x + 7$
- $(0,7),(2,30),(3,2),(4,-32) \implies Q = \frac{61}{24}x^3 - \frac{207}{8}x^2 + \frac{637}{12}x + 7$
- $(0,7),(2,30),(3,2),(5,-88) \implies Q = \frac{3}{2}x^3 - \frac{62}{3}x^2 + \frac{281}{6}x + 7$
- $(0,7),(2,30),(4,-32),(5,-88) \implies Q = \frac{11}{24}x^3 - \frac{107}{8}x^2 + \frac{437}{12}x + 7$
- $(0,7),(3,2),(4,-32),(5,-88) \implies Q = -\frac{7}{12}x^3 - 4x^2 + \frac{187}{12}x + 7$
- $(1,8),(2,30),(3,2),(4,-32) \implies Q = \frac{22}{3}x^3 - 69x^2 + \frac{533}{3}x - 108$
- $(1,8),(2,30),(3,2),(5,-88) \implies Q = \frac{29}{6}x^3 - 54x^2 + \frac{901}{6}x - 93$
- $(1,8),(2,30),(4,-32),(5,-88) \implies Q = \frac{7}{3}x^3 - 34x^2 + \frac{323}{3}x - 68$
- $(1,8),(3,2),(4,-32),(5,-88) \implies Q = -\frac{1}{6}x^3 - 9x^2 + \frac{211}{6}x - 18$
- $(2,30),(3,2),(4,-32),(5,-88) \implies Q = -\frac{8}{3}x^3 + 21x^2 - \frac{247}{3}x + 132$

Observăm că de fiecare dată a rezultat alt polinom. Prin urmare, deoarece fiecare polinom are câte un vot, nu avem garanția că algoritmul va determina corect polinomul folosit la codificare.

3.2 Algoritmul Berlekamp-Welch

Algoritmul *Berlekamp-Welch* este utilizat pentru decodarea și corectarea erorilor în cadrul codurilor Reed-Solomon și se bazează pe rezolvarea unui sistem supra-determinat de ecuații liniare. Comparativ cu metodele brute de interpolare, care pot avea o complexitate de $O(n^k \cdot k^2)$, acest algoritm reduce complexitatea la $O(n^3)$ prin utilizarea eliminării gaussiene.

Avem $k + 2s$ puncte (x_i, y_i), unde x_i sunt distincte, și dorim să determinăm care dintre acestea sunt eronate. Dacă am ști pozițiile erorilor, am putea folosi punctele corecte pentru a determina polinomul original $P(x)$, care are grad cel mult $k - 1$.

Pentru a modela erorile, folosim un polinom $E(x)$ astfel încât:

$$E(i) = 0 \iff \text{valoarea de la poziția } i \text{ este eronată.}$$

Dat fiind că valorile recepționate nu mai respectă relația ideală $y_i = P(x_i)$, dorim să determinăm două polinoame:

- $Q(x)$ - un polinom care înglobează atât mesajul original, cât și erorile;
- $E(x)$ - un polinom care indică pozițiile erorilor.

Acestea trebuie să satisfacă sistemul:

$$Q(x) = P(x) \cdot E(x), \tag{4}$$

$$Q(x_i) = y_i \cdot E(x_i), \quad \forall i \in \{0, 1, \ldots, k + 2s + 1\}. \tag{5}$$

Pentru determinarea acestor polinoame, ținem cont de:

- $Q(x)$ are gradul cel mult $k + s - 1$;
- $E(x)$ are gradul cel mult s, deci poate avea cel mult s rădăcini distincte.

Sistemul obținut este un sistem supra-determinat de ecuații liniare, cu $k + 2s$ ecuații și $k + 2s + 1$ necunoscute (coeficienții polinoamelor $Q(x)$ și $E(x)$). Acesta poate fi rezolvat eficient prin eliminare gaussiană, având complexitate $O(n^3)$.

Trebuie să menționăm că polinoamele Q și E sunt obținute prin eliminare gaussiana. După determinarea acestora, mesajul original este obținut prin împărțirea lui $Q(x)$ la $E(x)$, obținând $P(x)$. Dacă restul împărțirii este 0, înseamnă că mesajul a fost corect recuperat, permițând reconstrucția acestuia fără erori. În cazul în care restul nu este 0, acest lucru indică fie existența mai multor erori decât poate corecta algoritmul (adică mai mult de s erori), fie faptul că datele inițiale conțin erori care nu respectă ipoteza unui număr maxim de s erori corectabile. În acest caz, este posibil ca algoritmul să nu poată recupera corect mesajul original, iar alte metode de decodare ar trebui explorate.[Wik23]

3.2.1 Exemplu

Se consideră mesajul recepționat:

$$Y' = (7, 8, 10, 2, -27, -88)$$

în punctele:

$$X = (0, 1, 2, 3, 4, 5)$$

Polinomul original $P(x)$ este de grad cel mult 3, iar presupunem că există cel mult o eroare în date. Vom determina $P(x)$ folosind metoda Berlekamp-Welch.

Sistemul de ecuații
Presupunem existența a două polinoame:

- $N(x)$ de grad cel mult 4 (pentru că $N(x) = P(x)E(x)$)
- $E(x)$ de grad cel mult 1 (presupus sub forma $E(x) = x - e$)

Relația fundamentală este:

$$N(x_i) = y_i E(x_i), \quad \forall i \in \{0, 1, 2, 3, 4, 5\}$$

Dezvoltăm $N(x)$ și $E(x)$:

$$\begin{aligned} N(x) &= n_4x^4 + n_3x^3 + n_2x^2 + n_1x + n_0, \\ E(x) &= e_1x + e_0. \end{aligned}$$

Înlocuind în ecuațiile de mai sus, obținem sistemul:

$$\begin{aligned} n_4 \cdot 0^4 + n_3 \cdot 0^3 + n_2 \cdot 0^2 + n_1 \cdot 0 + n_0 &= 7e_0, \\ n_4 \cdot 1^4 + n_3 \cdot 1^3 + n_2 \cdot 1^2 + n_1 \cdot 1 + n_0 &= 8(e_1 + e_0), \\ n_4 \cdot 2^4 + n_3 \cdot 2^3 + n_2 \cdot 2^2 + n_1 \cdot 2 + n_0 &= 10(2e_1 + e_0), \\ n_4 \cdot 3^4 + n_3 \cdot 3^3 + n_2 \cdot 3^2 + n_1 \cdot 3 + n_0 &= 2(3e_1 + e_0), \\ n_4 \cdot 4^4 + n_3 \cdot 4^3 + n_2 \cdot 4^2 + n_1 \cdot 4 + n_0 &= -27(4e_1 + e_0), \\ n_4 \cdot 5^4 + n_3 \cdot 5^3 + n_2 \cdot 5^2 + n_1 \cdot 5 + n_0 &= -88(5e_1 + e_0). \end{aligned}$$

Rezolvarea sistemului
Rezolvând acest sistem, obținem soluțiile pentru coeficienții lui $N(x)$ în funcție de $E(x)$:

$$\begin{aligned} n_4 &= -\frac{11}{6}e_1, \\ n_3 &= -\frac{11}{6}e_0 + 6e_1, \\ n_2 &= 6e_0 - \frac{19}{6}e_1, \\ n_1 &= -\frac{19}{6}e_0 + 7e_1, \\ n_0 &= 7e_0. \end{aligned}$$

Dacă presupunem că eroarea a avut loc la $x = e$, putem verifica corectitudinea soluției cu datele primite și să determinăm eroarea.
Se obține polinomul corectat:

$$P(x) = -\frac{11}{6}x^3 + 6x^2 - \frac{19}{6}x + 7.$$

Astfel, am reconstituit mesajul corect.

3.2.2 De ce mergem cu numărul de erori de la mare la mic?

Un motiv fundamental pentru care nu începem de la un număr mic de erori este posibilitatea ca sistemul de ecuații rezultat să nu aibă soluții, deși acestea ar putea exista. Acest lucru se datorează faptului că un număr insuficient de termeni poate duce la o formulare incompletă a ecuațiilor necesare identificării erorilor.

Dacă începem de la un număr mare de erori, creștem probabilitatea de a găsi soluția, întrucât algoritmul parcurge inițial scenarii mai puțin probabile de a avea soluție, continuând spre cazuri mai probabile (mai puțin restrictive). În fiecare etapă a algoritmului, verificăm dacă restul împărțirii $Q(x)$ la $E(x)$ este zero:

$$R(x) = Q(x) \mod E(x), \tag{6}$$

unde $R(x) = 0$ indică faptul că am identificat un set valid de erori.

Procesul poate fi formalizat printr-un algoritm iterativ în care testăm succesiv grade mai mici ale polinomului erorilor $E(x)$:

1. Se inițializează un număr maxim t_{max} de erori posibile.
2. Pentru fiecare $t \in \{t_{max}, t_{max} - 1, \ldots, 0\}$:
   (a) Se generează ecuațiile corespunzătoare pentru erori.
   (b) Se verifică dacă restul $R(x) = 0$.
   (c) Dacă se obține $R(x) = 0$, se acceptă $E(x)$ ca soluție.
3. Dacă nu se găsește o soluție pentru niciun t, se consideră că numărul de erori depășește capacitatea de corecție a algoritmului.

În acest mod, evităm subestimarea numărului de erori și excludem scenariile în care soluția obținută ar fi incorectă din cauza presupunerii unui număr prea mic de erori. De asemenea, acest proces ajută la optimizarea resurselor computaționale, întrucât scenariile cu erori mai puține sunt mai probabile și sunt testate ulterior în mod controlat.

4 Concluzii

4.1 De ce am folosi aceste coduri și nu codurile Hamming?

Principalul motiv pentru care am alege să folosim codurile Reed-Solomon este posibilitatea corecției unui număr mare de erori, pe când în cadrul codurilor Hamming putem corecta un singur bit. Totuși, acestea vin și cu dezavantaje, ele fiind mai costisitoare de calculat și mai dificil de implementat.

Prin urmare, nu există un răspuns clar de care coduri sunt mai bune. Trebuie să alegem în funcție de situație: dacă vrem să suportăm corectarea mai multor erori vom alege Reed-Solomon, altfel Hamming.

4.2 Rezultate în practică

Algoritmul Berlekamp-Welch este mult mai eficient decât cel brut, decodificarea unui șir de date cu $k = 223$, $s = 16$ și fără erori durează doar $3,1$ secunde, folosind o implementare relativ naivă pentru eliminarea gaussiană, împărțirea de polinoame și alte precalculări.

Algoritmul se poate rula de asemenea în paralel, câte un block per thread, obținând astfel un timp și mai bun de rulare.

5 Anexe

5.1 Corpul Galois 256

Toate calculele făcute mai sus sunt făcute pe numere reale pentru a facilita înțelegerea conceptului. Totuși, știm foarte bine că aceste numere nu sunt cele mai potrivite când lucrăm pe un computer din pricina mai multor probleme printre care se numără:

1. Precizia finită
2. Erorile de precizie
3. Operațiile pe numere reale sunt mai încete decât cele pe numere întregi
4. Instabilitatea numerică

Pentru a remedia aceste probleme vom folosi un alt corp, mai exact corpul Galois 256, notat în continuare GF(256). Acest corp are niște proprietăți speciale care îl fac potrivit pentru scopul nostru, anume:

- Un element poate fi identificat printr-un număr de la 0 la 255, utilizând la maxim memoria sistemelor bazate pe octeți
- Operațiile pe acești identificatori sunt relativ rapide comparat cu operațiile pe numere reale

Definiție 1 *Un element al GF*(256) *este de fapt o clasă de resturi a polinoamelor cu coeficienți în* $\mathbb{Z}_2$*. Modulo-ul este un polinom ireductibil de grad* 8*.*

Câteva exemple de polinoame ireductibile ce pot fi folosite drept modulo sunt[Pla19]:

- $x^8 + x^4 + x^3 + x + 1$
- $x^8 + x^4 + x^3 + x^2 + 1$
- $x^8 + x^5 + x^3 + x + 1$
- $x^8 + x^5 + x^3 + x^2 + 1$
- $x^8 + x^5 + x^4 + x^3 + 1$

Deoarece vrem să mapăm bijectiv fiecare clasă de echivalență la octeți, vom proceda în sens invers, cel mai bine cu un exemplu:

- Octetul 0b00000000 va fi mapat la polinomul nul.
- Octetul 0b00000001 va fi mapat la polinomul 1.
- Octetul 0b00000010 va fi mapat la polinomul x.
- Octetul 0b00000011 va fi mapat la polinomul $x + 1$.
- Octetul 0b00000100 va fi mapat la polinomul x^2.
- Octetul 0b01010101 va fi mapat la polinomul $x^6 + x^4 + x^2 + 1$.
- Octetul 0b11111111 va fi mapat la polinomul $x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$.

În general, dacă al i-lea bit al octetului este setat atunci polinomul va conține termenul x^i.

Această mapare conferă o ușurință în ceea ce privește lucrul cu elementele corpului. Mai exact, suma a două polinoame se va transforma în operația de XOR logic. Înmulțirea cu x^i va fi echivalentă cu shiftarea la stânga a numărului cu i biți. Operația de înmulțire între două polinoame generale se poate aplica ușor folosind celelalte două operații.

Deoarece pentru operația de înmulțire trebuie să trecem prin toți biții unuia dintre operanzi iar apoi să reducem numărul dacă acesta reprezintă un polinom de grad mai mare de 7 putem precalcula toate înmulțirile și stoca rezultatele într-o matrice. Această matrice este de fapt "Tabla înmulțirii" pentru numerele noastre speciale.

Pentru operația de împărțire vom folosi un vector în care vom stoca la poziția i inversul polinomului corespunzător. Pentru împărțirea unui număr a la un alt număr b vom folosi următoarea formulă $a \otimes b^{-1}$

Caracteristica acestui corp este 2, ceea ce înseamnă că $p + p = 0, \forall p \in \text{GF}(256)$. Acest lucru se reflectă în operațiile pe octeți, $b \oplus b = 0$ ($\oplus$ = XOR logic). Datorită acestui fapt putem să nu precalculăm și opusul numărului, acesta fiind el însuși.

Reducerea unui polinom modulo $MOD = x^8 + x^4 + x^3 + x^2 + 1$.

Înmulțirea a două polinoame de grad ≤ 7 poate avea drept rezultat un polinom de grad > 8. Spre exemplu înmulțirea $(x^4+x^3+x+1)\cdot(x^6+x^2+1) = (x^{10}+x^9+x^7+x^6)+(x^6+x^5+x^3+x^2)+(x^4+x^3+x+1) = x^{10} + x^9 + x^7 + x^5 + x^4 + x^2 + x + 1$. Acest polinom nu este unul reprezentabil pe 8 biți, așa că îl vom reduce la un altul:

$$x^{10} + x^9 + x^7 + x^5 + x^4 + x^2 + x + 1 = (x^8 + x^4 + x^3 + x^2 + 1) \cdot x^2 + x^9 + x^7 + x^6 + x + 1 \equiv_{MOD} x^9 + x^7 + x^6 + x + 1$$

$$x^9 + x^7 + x^6 + x + 1 = (x^8 + x^4 + x^3 + x^2 + 1) \cdot x + x^7 + x^6 + x^5 + x^4 + x^3 + 1 \equiv_{MOD} x^7 + x^6 + x^5 + x^4 + x^3 + 1$$

Astfel am redus polinomul inițial de grad > 7 la unul de grad cel mult 7.

Aceste transformări sunt suficiente pentru a înlocui numerele reale de mai devreme cu GF(256). Astfel obținem un corp ale cărui operații sunt rapide pe procesor și nu au probleme cu precizia sau pierderea de informații din pricina operațiilor în sine.

Bibliografie

[Pla19] Cody Planteen. Interpolare lagrangiană, 2019. Accessed: 2025-01-30.

[Tr3] Petru Trîmbițaș. Eliminare gaussiană, 2013. Accessed: 2025-01-23.

[Ver22] Tom Verbeure. Reed-solomon error correcting codes from the bottom up, 2022. Accessed: 2025-01-23.

[Wik23] Wikipedia. Berlekamp-welch, 2023. Accessed: 2025-01-31.

[Wik24] Wikipedia. Interpolare lagrangiană, 2024. Accessed: 2025-01-24.

[Wik25] Wikipedia. Reed-solomon error correction, 2025. Accessed: 2025-01-23.