Coduri Reed-Solomon

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

Cuprins

Cuprins

Introducere

Introducere

Codurile Reed-Solomon reprezintă un procedeu de codificare a mesajelor care permite detectarea și corectarea erorilor care apar în transmiterea mesajului.
Dacă dorim să transmitem un mesaj de lungime k și să putem corecta cel mult s caractere greșite, atunci vom trimite un mesaj de dimensiune $k + 2s$.
Sunt folosite în principal la CD-uri, DVD-uri și QR Codes.

Codificarea

Codificarea

Există mai multe moduri de a realiza codificarea (și implicit decodificarea). În continuare vom vedea unul dintre cele mai simple moduri.

Codificarea

Există mai multe moduri de a realiza codificarea (și implicit decodificarea). În continuare vom vedea unul dintre cele mai simple moduri.
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 șir de octeți (sau mai comun simboluri).

Codificarea

Fie șirul $y_0, y_1, y_2, \ldots, y_{k-1}$.

Vom vrea să găsim un polinom special P de grad k, astfel încât $P(0) = y_0$, $P(1) = y_1$, $P(2) = y_2$, $\ldots$, $P(k-1) = y_{k-1}$.

Codificarea

Fie șirul $y_0, y_1, y_2, \ldots, y_{k-1}$.

Vom vrea să găsim un polinom special P de grad k, astfel încât $P(0) = y_0$, $P(1) = y_1$, $P(2) = y_2$, $\ldots$, $P(k-1) = y_{k-1}$.

După ce avem acest polinom vom calcula valorile $P(k)$, $P(k+1)$, $\ldots$, $P(k+2s-1)$.

În final vom stoca valorile
$y_0, y_1, y_2, \ldots, y_{k-1}, P(k), P(k+1), \ldots, P(k+2s-1)$.

Codificarea

Am introdus redundanță. Acum, bazat pe oricare k octeți îl putem găsi pe P, deci putem găsi $P(0) = y_0$, $P(1) = y_1$, $P(2) = y_2$, $\ldots$, $P(k-1) = y_{k-1}$.

Codificarea

Am introdus redundanță. Acum, bazat pe oricare k octeți îl putem găsi pe P, deci putem găsi $P(0) = y_0$, $P(1) = y_1$, $P(2) = y_2$, ..., $P(k-1) = y_{k-1}$.

Tot ce rămâne de făcut este să îl calculăm pe P.

Cum îl calculăm pe P?

Există două moduri principale de a afla polinomul P:

- Eliminare gaussiană
- Interpolare lagrangiană

Eliminare gaussiană

Vrem ca polinomul să respecte anumite condiții:

$$P(0) = y_0$$

$$P(1) = y_1$$

$$P(2) = y_2$$

$$\dots$$

$$P(k-1) = y_{k-1}$$

Eliminare gaussiană

Vrem ca polinomul să respecte anumite condiții:

$$
\begin{aligned}
&p_0 + p_1 \cdot 0 + p_2 \cdot 0^2 + \cdots + p_{k-1} \cdot 0^{k-1} = y_0 \\
&p_0 + p_1 \cdot 1 + p_2 \cdot 1^2 + \cdots + p_{k-1} \cdot 1^{k-1} = y_1 \\
&p_0 + p_1 \cdot 2 + p_2 \cdot 2^2 + \cdots + p_{k-1} \cdot 2^{k-1} = y_2 \\
&\vdots \\
&p_0 + p_1 \cdot (k-1) + p_2 \cdot (k-1)^2 + \cdots + p_{k-1} \cdot (k-1)^{k-1} = y_{k-1}
\end{aligned}
$$

Eliminare gaussiană

$$
\begin{aligned}
&p_0 + p_1 \cdot 0 + p_2 \cdot 0^2 + \cdots + p_{k-1} \cdot 0^{k-1} = y_0 \\
&p_0 + p_1 \cdot 1 + p_2 \cdot 1^2 + \cdots + p_{k-1} \cdot 1^{k-1} = y_1 \\
&p_0 + p_1 \cdot 2 + p_2 \cdot 2^2 + \cdots + p_{k-1} \cdot 2^{k-1} = y_2 \\
&\vdots \\
&p_0 + p_1 \cdot (k-1) + p_2 \cdot (k-1)^2 + \cdots + p_{k-1} \cdot (k-1)^{k-1} = y_{k-1}
\end{aligned}
$$

Aceste ecuații sunt suficiente pentru a afla polinomul P.

Interpolarea lagrangiană

Vom găsi polinomul P pe bucăți. Întâi vom vrea un polinom care are valoarea 0 în $x = 1$, $x = 2$, $x = 3$, $\ldots$, $x = k - 1$. Asta e ușor, e doar

$$Q_0(x) = (x - 1) \cdot (x - 2) \cdot (x - 3) \cdot \cdots \cdot (x - (k - 1)).$$

Interpolarea lagrangiană

Vom găsi polinomul P pe bucăți. Întâi vom vrea un polinom care are valoarea 0 în $x = 1$, $x = 2$, $x = 3$, $\ldots$, $x = k - 1$. Asta e ușor, e doar:

$$Q_0(x) = (x - 1) \cdot (x - 2) \cdot (x - 3) \cdot \cdots \cdot (x - (k - 1)).$$

Vrem acum un polinom care are valoarea 0 în $x = 1$, $x = 2$, $x = 3$, $\ldots$, $x = k - 1$ și valoarea y_0 în $x = 0$. Putem să alegem:

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

Interpolarea lagrangiană

Vom defini similar Q_n și R_n pentru $0 \leq n \leq k-1$. Avem :

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

Codificarea

Cele două metode determină același polinom (există un unic polinom de grad $k-1$ care să treacă prin cele k puncte).

Problema cu aceste metode este că cel mai probabil polinomul nu va avea coeficienți întregi. Pentru a rezolva asta vom folosi elemente din corpul $GF(256)$.

Decodificarea

Decodificarea

Acum că am codificat și stocat datele, vom vrea și un mod de a le decodifica și reobține chiar dacă diverse probleme au apărut în mediul de stocare.

Există mai multe modalități de decodificare.

Cea mai simplu de înțeles este metoda brută.

Decodificarea - brut

Nu putem spune exact care simboluri au fost modifiate. Putem totuși să facem mai multe încercări.

Alegem k elemente din setul de date. Generăm polinomul Q de grad $k-1$ care ar genera aceste puncte. Acest polinom primește un vot.

Decodificarea - brut

Nu putem spune exact care simboluri au fost modifiate. Putem totuși să facem mai multe încercări.

Alegem k elemente din setul de date. Generăm polinomul Q de grad $k-1$ care ar genera aceste puncte. Acest polinom primește un vot.

Repetăm pentru toate celelalte moduri de alegere a k elemente.

La final cel mai votat polinom este cel considerat corect (cel care a generat cele mai multe puncte).

Decodificarea - brut

Iterarea prin toate modurile de a alege k elemente din $k + 2s$ durează

$$\frac{(k + 2s)!}{k! \cdot (2s)!}.$$

Calcularea polinomului pentru un set de k elemente durează $O(k^3)$.
Metoda brută rulează în timp exponențial, mult prea mult pentru a putea fi folositoare în practică.

Exemplu - codificare

- Mesajul $y = (-1, 2, 3)$ pentru $x = (0, 1, 2)$.
- Polinomul $p = -x^2 + 4x - 1$.
- Evaluăm p în $x = 3$ și $x = 4$, deci știm că vom putea corecta cel mult o eroare.
- Mesajul transmis: $(-1, 2, 3, 2, -1)$.

Exemplu - decodificare

- Mesajul primit: $(-1, 2, 3, 4, -1)$.
- Toate posibilitățile de alegere sunt:
  - $(0, -1), (1, 2), (2, 3) \Rightarrow q = -x^2 + 4x - 1$
  - $(0, -1), (1, 2), (3, 4) \Rightarrow q = -\frac{2}{3}x^2 + \frac{11}{3}x - 1$
  - $(0, -1), (1, 2), (4, -1) \Rightarrow q = -x^2 + 4x - 1$
  - $(0, -1), (2, 3), (3, 4) \Rightarrow q = -\frac{1}{3}x^2 + \frac{8}{3}x - 1$
  - $(0, -1), (2, 3), (4, -1) \Rightarrow q = -x^2 + 4x - 1$
  - $(0, -1), (3, 4), (4, -1) \Rightarrow q = -\frac{5}{3}x^2 + \frac{20}{3}x - 1$
  - $(1, 2), (2, 3), (3, 4) \Rightarrow q = x + 1$
  - $(1, 2), (2, 3), (4, -1) \Rightarrow q = -x^2 + 4x - 1$
  - $(1, 2), (3, 4), (4, -1) \Rightarrow q = -2x^2 + 9x - 5$
  - $(2, 3), (3, 4), (4, -1) \Rightarrow q = -3x^2 + 16x - 17$

Exemplu - decodificare

- Mesajul primit: $(-1, 2, 3, 4, -1)$.
- Toate posibilitățile de alegere sunt:
  - $(0, -1), (1, 2), (2, 3) \Rightarrow q = -x^2 + 4x - 1$
  - $(0, -1), (1, 2), (3, 4) \Rightarrow q = -\frac{2}{3}x^2 + \frac{11}{3}x - 1$
  - $(0, -1), (1, 2), (4, -1) \Rightarrow q = -x^2 + 4x - 1$
  - $(0, -1), (2, 3), (3, 4) \Rightarrow q = -\frac{1}{3}x^2 + \frac{8}{3}x - 1$
  - $(0, -1), (2, 3), (4, -1) \Rightarrow q = -x^2 + 4x - 1$
  - $(0, -1), (3, 4), (4, -1) \Rightarrow q = -\frac{5}{3}x^2 + \frac{20}{3}x - 1$
  - $(1, 2), (2, 3), (3, 4) \Rightarrow q = x + 1$
  - $(1, 2), (2, 3), (4, -1) \Rightarrow q = -x^2 + 4x - 1$
  - $(1, 2), (3, 4), (4, -1) \Rightarrow q = -2x^2 + 9x - 5$
  - $(2, 3), (3, 4), (4, -1) \Rightarrow q = -3x^2 + 16x - 17$

Se observă că cel mai des apare polinomul $-x^2 + 4x - 1$, care este și cel corect.

Decodificarea - Berlekamp-Welch

Avem k+2s valori. Am vrea să aflăm care sunt valorile eronate dintre acestea (dacă am ști care sunt valorile eronate atunci le putem folosi pe cele bune pentru determinarea polinomului).

Decodificarea - Berlekamp-Welch

Avem k+2s valori. Am vrea să aflăm care sunt valorile eronate dintre acestea (dacă am ști care sunt valorile eronate atunci le putem folosi pe cele bune pentru determinarea polinomului).

Putem reprezenta valorile eronate printr-un polinom E. Mai exact, $E(i) = 0$ dacă și numai dacă valoarea de la poziția i nu este cea stocată inițial.

Decodificarea - Berlekamp-Welch

Vom nota cu $P(i)$ valoarea stocată inițial la indicele i și cu b_i elementul citit la indicele i. Relațiile dintre P, b și E sunt:

$$b_0 \cdot E(0) = P(0) \cdot E(0)$$

$$b_1 \cdot E(1) = P(1) \cdot E(1)$$

$$b_2 \cdot E(2) = P(2) \cdot E(2)$$

$$\vdots$$

$$b_{k-1} \cdot E(k-1) = P(k-1) \cdot E(k-1).$$

Decodificarea - Berlekamp-Welch

$$b_i \cdot E(i) = P(i) \cdot E(i)$$

Explicit, relația spune că fie $P(i) = b_i$, fie $E(i) = 0$, pentru fiecare i.

Decodificarea - Berlekamp-Welch

$$b_i \cdot E(i) = P(i) \cdot E(i)$$

Explicit, relația spune că fie $P(i) = b_i$, fie $E(i) = 0$, pentru fiecare i.
Notăm $F(i) = P(i) \cdot E(i)$.

Decodificarea - Berlekamp-Welch

$$b_i \cdot E(i) = P(i) \cdot E(i)$$

Explicit, relația spune că fie $P(i) = b_i$, fie $E(i) = 0$, pentru fiecare i.
Notăm $F(i) = P(i) \cdot E(i)$.

$$b_i \cdot \left(e_0 + e_1 \cdot i + e_2 \cdot i^2 + \cdots + e_{\text{err}} \cdot i^{\text{err}}\right) = f_0 + f_1 \cdot i + f_2 \cdot i^2 + \cdots + f_h \cdot i^h$$

(unde err este numărul de erori).

Decodificarea - Berlekamp-Welch

$$b_i \cdot \left(e_0 + e_1 \cdot i + e_2 \cdot i^2 + \cdots + e_{\text{err}} \cdot i^{\text{err}}\right) = f_0 + f_1 \cdot i + f_2 \cdot i^2 + \cdots + f_h \cdot i^h$$

În forma curentă, dacă ar fi să considerăm toate ecuațiile de această formă (necunoscutele sunt e_x și f_x) am avea prea multe necunoscute comparativ cu numărul de ecuații și deci nu am putea aplica eliminarea gaussiană.

Decodificarea - Berlekamp-Welch

$$b_i \cdot \left(e_0 + e_1 \cdot i + e_2 \cdot i^2 + \cdots + e_{\text{err}} \cdot i^{\text{err}}\right) = f_0 + f_1 \cdot i + f_2 \cdot i^2 + \cdots + f_h \cdot i^h$$

În forma curentă, dacă ar fi să considerăm toate ecuațiile de această formă (necunoscutele sunt e_x și f_x) am avea prea multe necunoscute comparativ cu numărul de ecuații și deci nu am putea aplica eliminarea gaussiană.

Putem totuși aplica un *hack*. Există multe polinoame care fac ce face E și acestea ne fac probleme. Așa că o să ne alegem o clasă restrânsă de posibilități pentru E.

Mai exact vom lua în considerare doar acele polinoame care au termenul dominant 1.

Decodificarea - Berlekamp-Welch

$$b_i \cdot \left(e_0 + e_1 \cdot i + e_2 \cdot i^2 + \cdots + e_{\text{err}} \cdot i^{\text{err}}\right) = f_0 + f_1 \cdot i + f_2 \cdot i^2 + \cdots + f_h \cdot i^h$$

Formula noastră devine, deci:

$$b_i \cdot \left(e_0 + e_1 \cdot i + e_2 \cdot i^2 + \cdots + e_{\text{err}-1} \cdot i^{\text{err}-1}\right) - \left(f_0 + f_1 \cdot i + f_2 \cdot i^2 + \cdots + f_h \cdot i^h\right)$$

Acum avem exact câte necunoscute ne trebuie și putem aplica eliminarea gaussiana.

Decodificarea - Berlekamp-Welch

Avem câteva cazuri posibile:

- Încă avem prea multă redundanță. Dacă ecuațiile sunt liniar dependente, asta înseamnă că numărul de erori considerate a fost prea mare. Deci putem repeta algoritmul, dar cu o valoare mai mică pentru variabila err.

Decodificarea - Berlekamp-Welch

Avem câteva cazuri posibile:

- Încă avem prea multă redundanță. Dacă ecuațiile sunt liniar dependente, asta înseamnă că numărul de erori considerate a fost prea mare. Deci putem repeta algoritmul, dar cu o valoare mai mică pentru variabila *err*.
- Obținem un rezultat reprezentat de coeficienții polinoamelor E și F.

Decodificarea - Berlekamp-Welch

În cazul al doilea, putem încerca să împărțim F la E. Dacă restul este polinomul nul, atunci câtul reprezintă polinomul P.

Decodificarea - Berlekamp-Welch

În cazul al doilea, putem încerca să împărțim F la E. Dacă restul este polinomul nul, atunci câtul reprezintă polinomul P.

Dacă însă avem rest nenul, am pierdut prea multă informație și datele originale nu mai pot fi reconstruite exact.

Decodificarea - Berlekamp-Welch

Acest algoritm are complexitatea $O\left((k+2s)^3 \cdot (s-\text{err})\right)$.

O particularitate a acestei metode de decodificare este că, dacă numărul de erori este mare, algoritmul rulează mai bine.

Concluzii

Concluzii

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.

Bibliografie

Bibliografie

- https://tomverbeure.github.io/2022/08/07/Reed-Solomon.htmlreed-solomon-encoding-through-polynomial-evaluation
- https://en.wikipedia.org/wiki/Lagrange_polynomial
- https://en.wikipedia.org/wiki/Reed-Solomon_error_correction
- https://en.wikipedia.org/wiki/Berlekamp-Welch_algorithm
- https://youtu.be/1pQJkt7-R4Q?si=z2IknmvGEYpIP4qT
- https://mathworld.wolfram.com/FiniteField.html
- https://codyplanteen.com/assets/rs/gf256_prim.pdf