Kody korekcyjne
notatki do przedmiotu

Edycja pierwsza 2019/20

Spis treści

Rozdział 1

Wstęp

Cel: wykrywania i poprawianie błędów:

- w zapisie
- transmisji
- przenoszeniu
- ...

Będziemy mówić o teorii kodów korekcyjnych. W szczególności model będzie dość obszerny.

Są kody robione ad-hoc i stosowane w różnych celach. My podamy konstrukcje: głównie algebraiczne (wielomiany i/lub przestrzenie liniowe). Będą to głównie kody binarne (lub dla potęg dwójki). Zastosowania wymagają odpowiedniego dobrania parametrów.

W zasadzie wszystkie działają przez pewną nadmiarowość.

Przykład 1.1. **kod Hamminga** ręczne wprowadzania bitów do komputera

kod IBAN chcemy tylko wykryć błędy (zamiana, zgubienie cyfry, przestawienie sąsiednich cyfr)

ISBN potrafi wykryć błąd, ale też można naprawić niewyraźną cyfrę (jedną)

CD/DVD często błędy występują seriami

korekcja w transmisji czy można poprosić o retransmisję?

Są dwa główne kierunki: najgorszy przypadek (model Hamminga) oraz statystyczny (model Shannona). Analiza jest różna, choć wiele konstrukcji używanych jest w jednym i drugim przypadku. Będziemy mówić w zasadzie wyłącznie o tym pierwszym modelu.

Książki i notatki:

- Venkatesan Guruswami, Atri Rudra i Madhu Sudan *Essential Coding Theory* `https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book/web-coding-book.pdf`
- San Ling, Chaoping Xing *Coding theory: a first course*
- Michael O'Sullivan *Coding Theory*, `https://mosullivan.sdsu.edu/Teaching/coding04.html`
- Mary Wootters, *Algebraic Error Correcting Codes*, `http://web.stanford.edu/~marykw/classes/CS250_W18/index.html`

Rozdział 2

Podstawy

Na podstawie [4, Rozdział 2].

Potrzebujemy modelu, który opisze zjawisko powstawania błędy.

Definicja 2.1. Kanał komunikacji (binary channel): jako wejście otrzymuje ciąg bitów, dla każdego z nich *niezależnie* przekazuje informację dalej, zmieniając ją z pewnym prawdopodobieństwem: $p_{ij} = \mathbb{P}[i \text{ na wyjściu} | j \text{ na wejściu}]$. Zakładamy, że $p_{1j} + p_{0j} = 1$ dla $j \in \{0, 1\}$.

Kanał jest *symetryczny*, jeśli $p_{10} = p_{01}$ oraz $p_{00} = p_{11}$, tj. prawdopodobieństwo błędu nie zależy od wejścia. Wtedy prawdopodobieństwo przekłamania oznaczamy jako γ.

Uwaga: czasami jako parametr używa się prawdopodobieństwa dobrej transmisji.

Dalej będziemy się zajmowali tylko i wyłącznie kanałami symetrycznymi, oznaczamy jako BSC_γ, przy czym rozważamy tylko $\gamma < 1/2$.

Zauważmy, że nasz kanał nie gubi informacji oraz nie „zaciera". Obsłużenie gubienia jest dość trudne. Z drugiej strony możemy rozszerzyć definicję o „zatarcie" (erasure), w której wyjściem może też być „$\perp$", oznaczający nieznany symbol. Kluczową różnicą z utratą bitu jest to, że wiemy, jaka jest długość wiadomości i na których miejscach występują bity.

Fakty poniżej w większości się uogólniają (w odpowiedni sposób) na kanały dopuszczające zatarcia.

Definicja 2.2. Mając dany kanał BSC_γ prawdopodobieństwo $\mathbb{P}[u|v]$ oznaczać będzie, że otrzymaliśmy u pod warunkiem, że słowem wejściowym jest v.

Dla danych u, v (tej samej długości) łatwo policzyć jest to prawdopodobieństwo.

Jeśli możliwe jest przesłanie każdej wiadomości, to nic nie wskóramy.

Definicja 2.3. *Kod*: $C \subseteq \Sigma^*$.

Kod *blokowy*: $C \subseteq \Sigma^n$ dla pewnego n.

Wielkość kodu $|C|$: liczba elementów w C.

Będziemy zajmować się głównie kodami blokowymi, tu dla prezentacji dla $\Sigma = \{0, 1\}$.

Przykład 2.4. **Kod powtórzeniowy** $w \to w^k$ dla pewnego k.

Kod Hamminga vel SECDED $(x_1, x_2, x_3, x_4) \in \{0, 1\}^4 \mapsto (x_1, x_2, x_3, x_4, x_2 + x_3 + x_4, x_1 + x_3 + x_4, x_1 + x_2 + x_3)$.

Bit kontroli parzystości $(x_1, x_2, \ldots, x_n) \in \{0, 1\}^n \mapsto (x_1, x_2, \ldots, x_n, \sum_{i=1}^n x_i)$.

Definicja 2.5 (Maximal probability decoding)**.** W Maximal probability decoding dla odebranego słowa u jako słowo kodowe odczytujemy v, takie że $\mathbb{P}[u|v]$ jest największe.

Są warianty, których w zasadzie nie będziemy rozróżniać: co jeśli parę słów jest równie prawdopodobnych? Możemy albo zwrócić dowolne z nich, albo zwrócić błąd/nic nie zwrócić.

Definicja 2.6. Odległość Hamminga.

Oznaczenie: $\mathrm{d}_H(u, v)$

Lemat 2.7. *Dla BSC_γ dla $\gamma < 1/2$, algorytm Maximal probability decoding zwraca słowo kodowe o najmniejszej odległości Hamminga.*

Dowód to proste rachunki do sprawdzenia na ćwiczeniach.

Definicja 2.8. Mówimy, że kod C wykrywa k błędów, jeśli dla słowa kodowego $v \in C$ na którym dokonamy k błędów algorytm Maximal probability decoding zwraca v lub nic nie zwraca. Mówimy, że kod C poprawia k błędów, jeśli w analogicznej sytuacji algorytm zawsze zwraca v.

Definicja 2.9. Odległość kodu C to $\mathrm{d_H}(C) = \min\limits_{u,v \in C, u \neq v} \mathrm{d_H}(u, v)$.

Lemat 2.10. *Kod C wykrywa k błędów wtedy i tylko wtedy, gdy $\mathrm{d_H}(C) < k$.*
Kod C poprawia k błędów wtedy i tylko wtedy, gdy $\mathrm{d_H}(C) < \left\lfloor \frac{k-1}{2} \right\rfloor$.

Dowód to proste rachunki.

Definicja 2.11. Kod blokowy C o słowach kodowych długości n nazywamy $(n, |C|, \mathrm{d_H}(C))$ kodem. Te liczby to parametry kodu.

Np. kod Hamminga, kod powtórzeniowy, bit parzystości.

Definicja 2.12. Zawartość informacyjna kodu $C \subseteq \Sigma^n$ (code rate, information rate):

$$R(C) = \frac{\log_{|\Sigma|} |C|}{n} = \frac{\log |C|}{n \log |\Sigma|}$$

Wymiar kodu (będzie miało sens potem):

$$\dim(R) = \log_{|\Sigma|} |C| = \frac{\log |C|}{\log |\Sigma|}$$

Zawartość informacyjna to intuicyjnie jaka frakcja pozycji („bitów") niesie informację. Sens wymairu stanie się jasny potem.

Rozdział 3

Kody liniowe

3.1 Kody jako przestrzenie liniowe

Na podstawie [4, Rozdział 4], [9, Rozdział 3].

Od teraz przez większość czasu zajmować się będziemy przypadkiem, że $\Sigma = \mathbb{F}$ to skończone ciało, zaś kod $C \subseteq \mathbb{F}^n$ to podprzestrzeń liniowa.

Definicja 3.1. Kod $C \subseteq \mathbb{F}^n$ nazywamy kodem liniowym, jeśli jest podprzestrzenią liniową $\mathbb{F}^n$.

Wymiar kodu ma teraz sens, bo to jest wymiar przestrzeni liniowej.

Taka reprezentacja ma wiele zalet:

- wiele dodatkowych własności
- łatwe kodowanie
- łatwiejsze dekodowanie
- konstrukcje

Przykład 3.2.

- kod Hamminga
- ISBN
- kody parzystości
- kody powtórzeniowe

Kod liniowy dla ciała o q elementach, długości słów kodowych n oraz wymiarze k oznaczamy przez $[n,k]_q$ kod. Jeśli jest odległość to d to oznaczamy go dodatkowo jako $[n,k,d]_q$. Oczywiście każdy $[n,k,d]_q$ kod jest też (n,q^k,d) kodem, ale niekoniecznie odwrotnie.

Lemat 3.3. *Dla kodu liniowego* C $\mathrm{d_H}(C) = \min_{0\neq v\in C} \|v\|_1$.

Dowód. Nierówność

$$\mathrm{d_H}(C) \leq \min_{0\neq v\in C} \|v\|_1$$

jest oczywista, bo

$$\min_{0\neq v\in C} \|v\|_1 = \min_{0\neq v\in C} d_H(0,v) \ .$$

W druga stronę:

$$\begin{aligned} \mathrm{d_H}(u,v) &= \mathrm{d_H}(0,u-v) \\ &= \|u-v\|_1 \end{aligned}$$

A dla $u,v \in C$ mamy też $u-v \in C$. □

Definicja 3.4 (Macierz generująca)**.** Dla kodu liniowego C jego macierz generujące G_C to dowolna macierz, której kolumny rozpinają C.

Przykład 3.5. **kod powtórzeniowy** Dla kodu powtórzeniowego o długości 6 macierzą generująca jest np.:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

kod Hamminga

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

kod parzystości Dla kodu parzystości, długości 4 macierzą generującą jest

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Macierz generująca służy do kodowania:

$$u \mapsto Gu$$

jest kodowaniem przy użyciu tej macierzy generującej.

Lemat 3.6. *Elementarne operacje kolumnowe przeprowadzone na macierzy generującej dają macierz generującą.*

Definicja 3.7. Mówimy, że dwa kody są *podobne*, jeśli jeden z nich można uzyskać przy użyciu operacji:

- przemnożenia wszystkich i-tych pozycji słów kodowych przez stałą różna od zera
- permutacji pozycji

Definicja 3.8 (Postać standardowa)**.** Macierz generujące jest w postaci standardowej, jeśli $G = \left[\frac{\mathrm{Id}}{M'}\right]$.

Taka postać ma zaletę, że do każdej wiadomości „dołączamy” na końcu dodatkowe bity.

Lemat 3.9. *Dla każdego kodu liniowego C istnieje podobny kod, który ma macierz generującą w postaci standardowej.*

Dowód. Przez operacje kolumnowe i wierszowe. □

3.2 Ogólny dekodowanie

Lemat 3.10. *Załóżmy, że słwo kodowe $v \in C$ zostało zmienione na v', przy czym $e = v' - v$ jest poprawialny, to jest $\|e\|_1 \leq \frac{d-1}{2}$. Wtedy e jest wektorem o minimalnej wadze Hamminga w warstwie $v' + C$ i jest on jedyny o takiej własności.*

Załóżmy, że po wysłaniu słowo kodowego v otrzymaliśmy v', Jeśli to prawda, to algorytm dekodujący sprawdza, w jakiej warstwie jest słowo i odejmuje taki wektor (zwany liderem warstwy).

Taki wektor nazywamy *liderem warstwy*. Niektóre warstwy (odpowiadające błędom, których nie potrafimy poprawić) mogą nie mieć liderów.

Dowód. Załóżmy, że w warstwie istnieje też e', t. że $\|e'\|_1 \leq \|e\|_1$. Wtedy $e' - e$ jest słowem kodowym. Zauważmy, że przesłanie $\vec{0}$ z błędem e' oraz $e' - e$ z błędem e daje ten sam komunikat e'. Czyli kod nie koryguje $\|e\|_1$ błędów, sprzeczność. □

3.2.1 Trudność ogólnego dekodowania

Zdefiniujmy problem Maximum likelyhood decoding of linear probblems (albo: closest codeword problem):
Dla danych $\vec{c} \in \mathbb{F}^n$ oraz macierzy generującej G kod C podaj $x \in C$, t. że $\mathrm{d_H}(\vec{c}, \vec{x})$ jest minimalne.

Może nie będziemy się rozwodzić nad wersją decyzyjną.

Ten problem jest NP-trudny. Jest też NP-trudny w aproksymacji z dowolnym stałym współczynnikiem, nawet dla $q = 2$.

Obliczanie odległości kodu też jest NP-trudne.

3.3 Kryptografia (schemat klucza publiczengo McEliece)

Klucze

- Bob wybiera binarny kod wymiaru k o długości słowa n, jako macierz G rozmiaru $n \times k$. Chcemy, aby kod *wydajnie* korygował t błędów.
- Bob losowo wybiera odwracalną macierz S o wyrazach z $\mathbb{F}$, rozmiaru $k \times k$.
- Bob wybiera losowo permutację n elementów, traktowaną jako macierz P rozmiaru $n \times n$.
- Klucz prywatny to (S, G, P).
- Klucz publiczny to $\widehat{G} = PGS$ oraz t.

Wysyłanie wiadomości:

- Wiadomość $x \in \mathbb{F}^n$
- Wybieramy losowo wektor t, $\|e\|_1 = t$.
- Wysyłamy $\widehat{G}x + e$

Dekodowanie

- Obliczamy $P^{-1}(\widehat{G}x + e) = GSe + \underbrace{P^{-1}e}_{e'}$, zauważmy, że $\|e'\|_1 = t$.
- Bob odtwarza Se (bo jest t błędów).

- Bob oblicza $x = S^{-1}Sx$.

Trudność: nie wiadomo, nie złamano — chcemy skonstruować najbliższe słowo kodowe dla kodu generowanego przez $\widehat{G}$.

3.4 Macierz parzystości

Reprezentowanie kodu przez macierz generującą jest zwykle dość kłopotliwe. Np. kody parzystości.

Zamiast tego możemy reprezentować przy użyciu równania. Albo ogólnie, macierzy parzystości.

Definicja 3.11. Dla kody C jego *macierz parzystości* to macierz H_C spełniająca:

$$C = \{v \,:\, H_C v = 0\} = \ker H_C$$

Uwaga. Macierz parzystości nie jest wyznaczona jednoznacznie. Czasami jedna postać jest istotnie lepsza, niż inna, np. podana poniżej macierz parzystości dla kodu Hamminga.

Przykład 3.12. **kod parzystości** Dla kodu parzystości, powiedzmy dla $n = 4$, macierz parzystości to:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

kod Hamminga Dla kodu Hamminga jego macierz parzystości to

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Zauważmy, że kolejne kolumny to zapis binarny kolejnych liczb naturalnych od 1 do 7.

Lemat 3.13. *Jeśli C ma wymiar k i słowa kodowe dł. n, to jego macierz parzystości H_C rząd* $\mathrm{rk}(H_C) = n - k$.

Bez zmniejszenia ogólności możemy założyć, że wiersze macierzy parzystości są liniowo niezależne.

Lemat 3.14. *Dla kodu C jego odległość* $\mathrm{d_H}(C)$ *to minimalna (dodatnia) liczba kolumn liniowo zależnych w macierzy parzystości.*

Lemat 3.15. *Jeśli macierz generatorów jest w postaci standardowej $G_C = (\mathrm{Id}\,|X)$ to macierz parzystości jest postaci*

$$\begin{bmatrix} -X \\ \mathrm{Id} \end{bmatrix}$$

Dowód to prosty rachunek, który pozostawiamy na ćwiczenia.

Uwaga, taka postać macierzy parzystości niekoniecznie jest najlepsza...

3.4.1 Iloczyn skalarny

Definicja 3.16. Dla przestrzeni liniowej $\mathbb{F}^k$ jej *standardowym iloczynem skalarnym* nazywamy:

$$\left\langle [a_1, \ldots, a_k]^T, [b_1, \ldots, b_k]^T \right\rangle = \sum_{i=1}^{k} a_i \cdot b_i \ .$$

Lemat 3.17. *Iloczyn skalarny jest funkcjonałem dwuliniwoym, tj. po ustaleniu jednego argumentu jest funkcją liniową.*

Iloczyn skalarny jest symetryczny (od argumentów).

Definicja 3.18. Mówimy, że dwa wektory są *prostopadłe*, co oznaczamy jako $u \perp v$, jeśli $\langle u, v\rangle = 0$. *Dopełnienie ortogonalne* zbioru W to

$$\{v \,:\, \forall_{w\in W} v \perp w\}$$

Uwaga. Uwaga, C oraz $C^\perp$ nie muszą być rozłączne! Może się nawet zdarzyć, że $C = C^\perp$.

Uwaga. Wektor może być prostopadły sam do siebie, np.

$$\left\langle [1,1]^T, [1,1]^T \right\rangle = 1 + 1 = 0$$

Lemat 3.19. *Dla zbioru W: $W^\perp$ jest przestrzenią liniową. Jeśli W jest podprzestrzenią liniową*

- *wymiar $W^\perp$ to $n - \dim(W)$*
- $(W^\perp)^\perp = W$.

3.5 Kody dualne

Definicja 3.20. Dla kodu C kod $C^\perp$ nazywamy kodem dualnym.

Kod nazywamy *samo-prostopadłym*, jeśli $C \subseteq C^\perp$.

Kod nazywamy *samo-dualnym*, jeśli $C = C^\perp$.

Lemat 3.21. *Dla kodu C macierz H_C^T jest macierzą generującą $C^\perp$.*

3.6 Dekodowanie na podstawie syndromu

Definicja 3.22. $H_C v$ to *syndrom* v.

Zauważmy, że $\vec{0}$ wychodzi wtedy i tylko wtedy, gdy $v \in C$.

Lemat 3.23. *Jeśli H_C jest macierzą parzystości, to syndrom $H_C v$ jet taki sam na całej warstwie.*

Syndromy są też różne dla różnych warstw.

W szczególności są różne dla różnych błędów, które potrafi poprawić kod.

Dowód.

$$H_C(C + e) = H_C C + H_C e = \vec{0} + H_C e$$

$H_C v = H_C w$ implikuje, że $H_C(v - w) = 0$, czyli $v - w \in \ker H_C = C$.

Załóżmy, że dla poprawialnych błędów e_1, e_2 mamy, że ich syndromy sa takie same. Ale to oznacza, że są w tej samej warstwie, co już wykluczyliśmy w Rozdziale 3.2. □

Mapowanie $H_C e \mapsto e$ nazywa się w literaturze *Syndrome lookup table* i jest jedną z praktycznych metod dekodowania kodów, o ile błędów jest mało.

Co tu dokładnie trzeba pokazać?

3.7 Kody Hamminga dokładniej

3.7.1 Dekodowanie na postawie syndromu

Niech u zostanie przekazane jako v. Oznaczmy $e = v - u$, jest to błąd.

Rozpatrzmy

$$H_C v = H_C(G_C u + e) = \vec{0} + H_C e$$

Jeśli e jest pojedynczym błędem, to $H_C e = H_C v$ to numer tego błędu (bitu, który trzeba usunąć).

3.7.2 Uogólnione kody Hamminga

Niech H_r to macierz rozmiaru $r \times (2^r - 1)$, której i-ta kolumna to binarny zapis i. Zdefiniujmy $C^{(r)}_{\mathrm{Ham}}$ jako kod, którego macierz parzystości ma zapis binarny i (przy użyciu r bitów) jako i-tą kolumnę, dla $i = 1, \ldots, 2^r - 1$.

Lemat 3.24. *$C^{(r)}_{Ham}$ to $[2^r - 1, 2^r - 1 - r, 3]_2$ kod. W szczególności poprawia on jeden błąd.*

Dowód. Długość kodu to liczba kolumn macierzy parzystości, czyli $2^r - 1$ (dla zwykłego kodu Hamminga: $2^3 - 1 = 7$). Wymiar to $2^r - 1$ minus rząd macierzy parzystości, czyli r (np. dla najprostszego kodu Hamminga: $2^3 - 1 - 3 = 4$).

Co do odległości, to łatwo zauważyć, że nie ma dwóch kolumn liniowo zależnych w macierzy parzystości, zaś istnieją 3 (łatwo wybrać). □

Algorytm dekodujący jest jak dla zwykłego kodu Hamminga.: używamy syndromów, dają numer bitu do poprawienia.

3.7.3 Kody Hadamarda (kody simplex)

Gdy C jako kod ma dobre własności, to $C^{\perp}$ też. To jest fakt empiryczny, nie twierdzenie.

Kody Hadamarda to (prawie) kody dualne do kodów Hamminga.

Kody simplex: kody dualne do kodów Hamminga. Kody Hadamarda: dodajemy do kodów simpleks wiersz zerowy (czyli ten sam kod + bit 0 na samym końcu).

Kod simplex to $[2^r - 1, r]$ kod a kod Hadamarda to $[2^r, r]$ kod.

Had_r.

Kod Hadamarda to najbardziej nadmiarowy kod, jak to tylko możliwe: koduje słowo długości r przy użyciu iloczynu skalarnego z każdym słowem długości r (więcej się po prostu nie da), tj. przy użyciu ciągu długości 2^r. Więcej się po prostu nie da.

Lemat 3.25. $\mathrm{d_H}(\mathrm{Had}_r) = 2^{r-1}$*, tak samo dla kodu.*

Dowód. Policzymy, jaka jest minimalna liczba jedynek w niezerowym słowie kodowym.

Zauważmy, że słowo kodowe uzyskujemy przez zapisanie 2^r iloczynów z każdym możliwym wektorem długości r. Łatwo sprawdzić, że dokładnie połowa takich iloczynów to 0, zaś połowa to 1: jeśli słowo kodowe ma 1 na j-tej pozycji, to

$$\langle w, v \rangle = 1 + \langle w, v + e_j \rangle$$

zaś dodanie e_j to bijekcja między zbiorem wektorów długości r. □

Rozdział 4

Ograniczenia kombinatoryczne

Na podstawie [8, Rozdział 4], [4, Rozdział 5] oraz [9, Wykład 3, 4].

4.1 Ograniczenie Hamminga, kody doskonałe

Twierdzenie 4.1 (Ograniczenie Hamminga). *Jeśli C jest $(n, |C|, d)_q$ kodem, to*

$$|C| \cdot \sum_{i=0}^{\lfloor \frac{d-1}{2} \rfloor} \binom{n}{i} (q-1)^i \leq q^n \ .$$

Równoważnie, zawartość informacyjna takiego kodu to najwyżej:

$$R(C) \leq 1 - \frac{\log_q \left(\sum_{i=0}^{\lfloor \frac{d-1}{2} \rfloor} \binom{n}{i} q^i \right)}{n} \ .$$

Definicja 4.2. $[n, k, d]_q$ kod nazywamy *doskonałym*, jeśli każde słowo w $\mathbb{F}_q^n$ jest w odległości najwyżej $\left\lfloor \frac{d-1}{2} \right\rfloor$ od jakiegoś słowa kodowego.

Innymi słowy, nic się nie marnuje, tj. ograniczenie Hamminga jest ścisłe.

Fakt 4.3. *Jeśli $[n, k, d]_q$-kod jest nietrywialny (tj. nie jest całą przestrzenią, ani nie jest jednoelementowy), to d jest nieparzyste.*

Dowód pozostawiamy jako proste ćwiczenie.

Twierdzenie 4.4 (van Lint, Tietaveinen). *Binarny kod doskonały jest jednej z następujących postaci:*

- *uogólniony kod Hamminga $C_{Ham}^{(r)}$.*
- *Kod Golaya [23, 12, 7].*
- *Nieparzyste powtórzenia jednego znaku, tj. kod $\{0^n, 1^n\}$ dla nieparzystego n.*
- *trywialne kody (tj. cała przestrzeń lub jedno słowo kodowe)*

Dowód. Dowód jest w ogólności trudny.

Pokażemy, że kod Hamminga jest doskonały. W tym celu wystarczy pokazać, że każde słowo spoza kodu jest w odległości 1 od słowa kodowego. Niech H_r będzie macierzą parzystości. Rozważmy słowo w nie będąc słowem kodowym. Hw to niezerowy wektor, czyli jest on którąś kolumną H (bo H_r zawiera wszystkie niezerowe kolumny), powiedzmy j-tą. Wtedy $H_r(w+e_j) = \vec{0}$, i w takim razie $w + e_j$ jest słowem kodowym.

Dowód można też przeprowadzić przez bezpośrednie zliczanie w nierówności Hamminga, jest to jednak trochę uciążliwe. □

4.2 Trochę oznaczeń

Definicja 4.5.

$$A_q(n,d) = \max\{M : \text{ istnieje } (n,M,d)_q \text{ kod}\}$$

Kod spełniający takie ograniczenie to *kod optymalny*.

$$B_q(n,d) = \max\{q^k : \text{ istnieje } [k,M,d]_q \text{ kod}\}$$

Twierdzenie 4.6. *1.* $B_q(n,q) \leq A_q(n,d) \leq q^n$ *dla* $1 \leq d \leq n$

2. $B_q(n,1) = A_q(n,1) = q^n$

3. $B_q(n,n) = A_q(n,n) = q$

Definicja 4.7 (Kod rozszerzony)**.** Dla kodu C jego kod rozszerzony powstaje przez dopisanie bitu kontroli parzystości, formalnie:

$$\overline{C} = \{(c_1,\ldots,c_k,-\sum_{i=1}^{k} c_i) \ : \ c_1,\ldots,c_k \in C\}$$

Ten przepis mówi, jak powstaje macierz generatorów. Macierz parzystości powstaje przez dodanie do każdego wiersza 0 na końcu oraz wiersza samych jedynek.

Lemat 4.8.

$$\mathrm{d_H}(C) \leq \mathrm{d_H}(\overline{C}) \leq \mathrm{d_H}(C) + 1$$

Oba ograniczenia są realizowalne.

Twierdzenie 4.9. *Jeśli d jest nieparzyste, to:*

1. $(n,M,d)_2$*-kod istnieje* $\iff$ $(n+1,M,d+1)_2$ *kod istnieje. W szczególności*

$$A_2(n+1,d+1) = A_2(n,d)$$

2. $[n,k,d]_2$*-kod istnieje* $\iff$ $[n+1,k,d+1]_2$ *kod istnieje. W szczególności*

$$B_2(n+1,d+1) = B_2(n,d)$$

Prosty dowód pozostawimy jako ćwiczenie.
Przeformułowanie ograniczenia Hamminga:

$$A_q(n,d) \leq q^n / \sum_{i=0}^{\lfloor \frac{d-1}{2} \rfloor} \binom{n}{i} (q-1)^i \ .$$

4.3 Ograniczenie (dolne) Gilbert-Vershamov

Twierdzenie 4.10 (Ograniczenie (dolne) Gilbert-Vershamov: wersja Gilberta, vel pokrywanie sferami)**.**

$$A_q(n,d) \geq q^n / \sum_{i=0}^{d} \binom{n}{i} (q-1)^i \ .$$

Dowód. Weźmy dowolny kod optymalny. Nie ma wektora, który jest w odległości większej niż d od każdego słowa kodowego: takie słowo można wziąć.

Czyli suma $A_q(n,d)$ sfer o promieniu $d-1$ pokrywa całą przestrzeń:

$$A_q(n,d) \cdot \sum_{i=0}^{d} \binom{n}{i} (q-1)^i \geq q^n \quad . \quad \square$$

W nielicznych przypadkach to daje ostre ograniczenia.

Twierdzenie 4.11 (Ograniczenie (dolne) Gilbert-Vershamov: wersja Verschamova)**.** *Niech n, k, d: liczby całkowite spełniające $2 \leq d \leq n$, $1 \leq k \leq n$. Jeśli*

$$\sum_{i=0}^{d-2} \binom{n-1}{i} (q-1)^i \leq q^{n-k}$$

to istnieje $[n, k, d']_2$ kod dla pewnego $d' \geq d$.

To ograniczenie ma też wersję asymptotyczną i długo nieznane były kody, które to asymptotyczne ograniczenie poprawiają. Nie wiadomo, czy istnieją kody binarne, które są lepsze niż to ograniczenie.

Dowód. Stworzymy macierz parzystości rozmiaru $(n-k) \times n$, taką że dowolne $d-1$ jej kolumn jest liniowo niezależnych. Kod takiej macierzy spełnia warunki zadania.

Pierwszą kolumnę wybieramy dowolnie. Następnie i-tą kolumnę wybieramy dowolnie spośród dostępnych q^{n-k} kolumn, jednak tak, by nie była w podprzestrzeni rozpostartej przez dowolne $d-1$ kolumn stworzonej już macierzy. Jest to możliwe, bo

$$\underbrace{\sum_{i=0}^{d-2}}_{\substack{i \text{ — liczba niezerowych wektorów}}} \quad \underbrace{\binom{j-1}{i}}_{\substack{\text{wybór wektorów do kombinacji}}} \quad \underbrace{(q-1)^i}_{\substack{\text{niezerowe współczynniki przy wektorach}}} \leq$$

$$\leq \sum_{i=0}^{d-2} \binom{n-1}{i} (q-1)^i < q^{n-k} \quad \square$$

W istocie, *losowy* kod spełnia ograniczenie bliskie ograniczenie GV.

Uwaga. Ograniczenie GV pozostaje niepoprawione od lat 70., wiele lat później pokazano istotnie lepsze kody (oparte na geometrii algebraicznej) dla $q \geq 49$ (kody te działają dla kwadratów liczb pierwszych, dopiero dla 7 i większych liczb dają coś lepszego niż ograniczenie GV).

W przyppadku kodów binarnych znane są jedynie bardzo dorbne poprawki w bardzo szczególnych przypadkach — duży problem otwarty.

4.4 Ograniczenie (górne) Singleton'a

Twierdzenie 4.12 (Ograniczenie Singletona)**.**

$$A_q(n,d) \leq q^{n-d+1}$$

Równoważnie:

$$R(C) \leq \frac{n-d+1}{n}$$

Kody, dla których ta nierówność jest spełniona z równością, nazywamy *MDS-kodami* (*maximal distance separability*). Równoważnie możemy powiedzieć, że są to kody, dla których $\mathrm{d_H}(C) = n-k+1$.

Dowód. Weźmy dowolny kod C spełniający $\mathrm{d_H}(C) \leq d$. Usuńmy ostatnich $d-1$ współrzędnych. Ze względu na odległość, pozostałe słowa są wciąż różne i mają $n-d+1$ współrzędnych, czyli

$$|C| \leq q^{n-d+1}$$

co daje tezę. □

Uwaga. Wersję ograniczenia dla kodów liniowych mówi po prostu, że dla $n, k, d]_q$ kodu zachodzi:

$$k + d \leq n + 1$$

Tę nierówność można dość prosto pokazać.

Twierdzenie 4.13. *Niech C będzie $[n, k, d]_q$ kodem liniowym. Niech G, H będą, odpowiednio, jego macierzami generującymi oraz macierzą parzystości. Następujące warunki są równoważne:*

1. *C jest MDS*
2. *każde $n-k$ kolumn H jest liniowo niezależnych*
3. *każde k wierszy G jest liniowo niezależnych*
4. *$C^\perp$ jest MDS*

Dowód: ćwiczenie.

Definicja 4.14. MDS kod C jest trywialny, jeśli jest w jednej z poniższych postaci:

- $C = \mathbb{F}_q^n$
- C jest generowany przez $(1, \ldots, 1)^T$
- C jest dualny do powyższego kodu.

W przeciwnym razie, C jest *nietrywialny.*

Można też rozpatrywać kod dualny do $\mathbb{F}_q^n$, ale ciężko w nim liczyć odległość.

Uwaga. Dla $q = 2$ istnieją tylko trywialne kody MDS. Dowód pozostawiamy jako (proste) ćwiczenie.

4.5 Ograniczenie (górne) Plotkina

Dla tych, którzy słyszeli nazwisko Plotkina — to inny Plotkin.

Twierdzenie 4.15 (Ograniczenie Plotkina)**.** *Jeśli $\frac{q-1}{q}n < d$ to*

$$A_q(n, d) \leq \left\lfloor \frac{d}{d - \frac{q-1}{q}n} \right\rfloor \quad .$$

Dowód. Niech

$$T = \sum_{c,c'} \mathrm{d_H}(c, c')$$

Jako że $d \leq d(c, c')$ dla każdych $c, c' \in C$, dostajemy

$$M(M-1)d \leq T$$

Oszacujemy teraz T z drugiej strony. Zauważmy, że jest ono równe:

$$T = \sum_{i=1}^{n} \sum_{c,c' \in C} d(c_i, c'_i)$$

Spróbujmy oszacować drugą sumę. Nie $n_{i,a}$ to liczba słów kodowych, w których i-ta współrzędna to a. Oczywiście

$$\sum_{a \in \Sigma} n_{i,a} = M$$

Wtedy

$$\sum_{c,c' \in C} d(c_i, c'_i) = \sum_{a \in \Sigma} n_{i,a} \cdot (M - n_{i,a})$$

(Ustalamy literę, potem bierzemy słowo, która ma taką literę i słowo, które nie ma).
Czyli

$$\begin{aligned} T &= \sum_{i=1}^{n} \sum_{c,c' \in C} d(c_i, c'_i) \\ &= \sum_{i=1}^{n} \sum_{a \in \Sigma} n_{i,a} \cdot (M - n_{i,a}) \\ &= M^2 n - \sum_{i=1}^{n} \sum_{a \in \Sigma} n_{i,a}^2 \end{aligned}$$

Teraz używając jednej z ulubionych średnich, np. C-S albo Jensena, albo...

$$\begin{aligned} M^2 n - \sum_{i=1}^{n} \sum_{a \in \Sigma} n_{i,a}^2 &\leq M^2 n - \sum_{i=1}^{n} \frac{\left(\sum_{a \in \Sigma} n_{i,a}\right)^2}{q} \\ &= M^2 n - \sum_{i=1}^{n} \frac{M^2}{q} \\ &= \frac{q-1}{q} M^2 n \end{aligned}$$

Porównując oba ograniczenia dostajemy:

$$\begin{aligned} M(M-1)d \leq \frac{q-1}{q} M^2 n &\iff (M-1)d \leq \frac{q-1}{q} Mn \\ &\iff M\left(d - \frac{q-1}{q} n\right) \leq d \qquad \text{przy założeniu } d > \frac{q-1}{q} n \\ &\iff M \leq \frac{d}{\left(d - \frac{q-1}{q} n\right)} \qquad \square \end{aligned}$$

Ograniczenie Plotkina można trochę poprawić, dla $q = 2$ (ćwiczenie). Czasami ograniczeniem Plotkina nazywa się taką właśnie wersję.

Twierdzenie 4.16 (Ograniczenie Plotkina dla kodów binarnych)**.** *Jeśli d jest parzyste, to*

$$A_2(n,d) \leq \begin{cases} 2\left\lfloor \frac{d}{2d-n} \right\rfloor & , \text{ dla } n < 2d \\ 4d & , \text{ dla } n = 2d \end{cases}.$$

Jeśli d jest nieparzyste, to

$$A_2(n,d) \leq \begin{cases} 2\left\lfloor \frac{d+1}{2d+1-n} \right\rfloor & , \text{ dla } n < 2d+1 \\ 4d+4 & , \text{ dla } n = 2d+1 \end{cases}.$$

Dla przypadku $d = \frac{q-1}{q}n$ można podać inne ograniczenie, zwykle nie nazywane jednak ograniczeniem Plotkina. Dowód jest zupełnie inny.

Twierdzenie 4.17 (Uogólnienie ograniczenia Plotkina)**.** *Niech $d = \frac{q-1}{q}n$. Wtedy*

$$A_q(n,d) \leq 2qn.$$

Dowód. Zdefiniujmy (współrzędne w $\mathbb{R}^q$ liczymy od 0 do $q-1$)

$$\varphi : \mathbb{F}_q \to \mathbb{R}^q, \quad \varphi(i) = \frac{1}{q}(1,1,\ldots,1,\underbrace{-(q-1)}_{i\text{-ta pozycja}},1,\ldots,1)^T = \frac{1}{q}(1,\ldots,1)^T - \vec{E_i}$$

oraz

$$f : C \to \mathbb{R}^{nq} \quad f(c) = \underbrace{\sqrt{\frac{q}{n(q-1)}}}_{\text{czynnik skalujący}} (\varphi(c_1),\ldots,\varphi(c_n))^T$$

Łatwo sprawdzić, że:

- f jest różnowartościowe
- $\langle f(c), f(c')\rangle = 1 - \frac{q}{q-1}\frac{\mathrm{d_H}(c,c')}{n}$ dla $c, c' \in C$

Stosując tę funkcję do naszego kodu otrzymujemy M wektorów w $\mathbb{R}^{nq}$ długości 1, takich że dla $v \neq v'$ mamy

$$\begin{aligned}\langle v, v'\rangle &\leq 1 - \frac{q}{q-1}\frac{d}{n} \\ &= 0\end{aligned}$$

Jako prosty lemat pozostawiamy udowodnienie, że taki układ ma najwyżej $2nq$ wektorów, (oszacowanie na liczbę wektorów jest ścisłe). Czyli $M \leq 2nq$, co daje to oszacowanie. □

Uwaga. To podejście daje też dowód w ogólnym przypadku ograniczenia Plotkina, ale dowód jest dość techniczny.

4.6 Asymptotyka objętości kul

Chcemy oszacować objętość kuli o promieniu r w $\mathbb{F}_q^n$

$$\mathrm{Vol}_q(r,n) = \sum_{i=0}^{r}\binom{n}{r}(q-1)^i$$

Definicja 4.18 (q-ta entropia)**.** Zdefiniujmy

$$H_q(x) = x\log_q(q-1) - x\log_q(x) - (1-x)\log_q(1-x)$$

Dla $q = 2$ to jest funkcja entropii Shannona, dla większych to ciężko o taką interpretację. Nie ma też prostego związku z entropią empiryczną dla większej liczby liter. Maksymalna wartość to 1 dla $x = \frac{q-1}{q}$.

Zwykle bedziemy szacowali $q^{H_q(p)n}$, policzmy, ile wynosi

$$q^{H_q(p)n} = \frac{(q-1)^{pn}}{p^{pn}\cdot(1-p)^{(1-p)n}} \ .$$

Lemat 4.19. *Dla całkowitego $q \geq 2$ oraz naturalnego p spełniającego $0 \leq p \leq \frac{q-1}{q}$ zachodzi*

$$\begin{aligned} \mathrm{Vol}_q(\lfloor pn \rfloor, n) &\leq q^{H_q(p)n} \\ \mathrm{Vol}_q(pn, n) &\geq q^{H_q(p)n - o(n)} \qquad \textit{dla dostatecznie dużego } n \end{aligned}$$

Uwaga. Formalnie powinny być jakieś sufity i/lub podłogi, ale stosujemy tylko w wypadku, gdy to jest liczba naturalna.

Dowód.

$$\begin{aligned} 1 &= (p + (1-p))^n \\ &= \sum_{i=0}^{n} \binom{n}{i} p^i (1-p)^{n-i} \\ &\geq \sum_{i=0}^{\lfloor pn \rfloor} \binom{n}{i} p^i (1-p)^{n-i} \\ &= \sum_{i=0}^{\lfloor pn \rfloor} \binom{n}{i} (q-1)^i \left(\frac{p}{q-1}\right)^i (1-p)^{n-i} \\ &= \sum_{i=0}^{\lfloor pn \rfloor} \binom{n}{i} (q-1)^i \left(\frac{p}{(q-1)(1-p)}\right)^i (1-p)^{n} \end{aligned}$$

skoro $p \leq \frac{q-1}{q}$ to $\frac{p}{(q-1)(p-1)} \leq 1$

$$\begin{aligned} \sum_{i=0}^{\lfloor pn \rfloor} \binom{n}{i} (q-1)^i \left(\frac{p}{(q-1)(1-p)}\right)^i (1-p)^n &\geq \sum_{i=0}^{\lfloor pn \rfloor} \binom{n}{i} (q-1)^i \left(\frac{p}{(q-1)(1-p)}\right)^{pn} (1-p)^n \\ &= \left(\frac{p}{(q-1)(1-p)}\right)^{pn} (1-p)^n \cdot \sum_{i=0}^{\lfloor pn \rfloor} \binom{n}{i} (q-1)^i \\ &= \underbrace{\frac{p^{pn}(1-p)^{(1-p)n}}{(q-1)^{pn}}}_{q^{-H_q(p)n}} \cdot \underbrace{\sum_{i=0}^{\lfloor pn \rfloor} \binom{n}{i} (q-1)^i}_{\mathrm{Vol}_q(\lfloor pn \rfloor, n)} \end{aligned}$$

Do drugiej części będziemy potrzebowali oszacowania Stirlinga:

$$\sqrt{2\pi n} \left(\frac{n}{e}\right)^n e^{1/(12n+1)} < n! < \sqrt{2\pi n} \left(\frac{n}{e}\right)^n e^{1/(12n)}$$

Oznaczmy $\lambda_1(n) = \frac{1}{12n}$, $\lambda_2(n) = \frac{1}{12n+1}$

$$\begin{aligned} \binom{n}{\lfloor pn \rfloor} &= \frac{n!}{\lfloor pn \rfloor! (n - \lfloor pn \rfloor)!} \\ &> \frac{(n/e)^n}{(\lfloor pn \rfloor / e)^{\lfloor pn \rfloor} / ((n - \lfloor pn \rfloor)/e)^{(n - \lfloor pn \rfloor)}} \cdot \frac{1}{\sqrt{2\pi p(1-p)n}} \cdot e^{\lambda_1(n) - \lambda_2(pn) - \lambda_2((1-p)n)} \\ &= \frac{1}{p^{pn}(1-p)^{(1-p)n}} \cdot \frac{1}{\sqrt{2\pi p(1-p)n}} \cdot e^{\lambda_1(n) - \lambda_2(pn) - \lambda_2((1-p)n)} \end{aligned}$$

I teraz

$$\begin{aligned}\mathrm{Vol}(pn,n) &\geq \binom{n}{pn}(q-1)^{pn} \\ &> \frac{(q-1)^{pn}}{p^{pn}(1-p)^{(1-p)n}} \cdot \frac{1}{\sqrt{2\pi p(1-p)n}} \cdot e^{\lambda_1(n)-\lambda_2(pn)-\lambda_2((1-p)n)} \\ &\geq e^{H_q(n)n-o(n)}\end{aligned}$$

□

Rozdział 5

Kody Reeda Solomona

Na podstawie [8, Rozdział 5, Rozdział 15] oraz [9, Wykład 4, 5].

5.1 Kody Reeda-Solomona

Definicja 5.1. Ustalmy ciało $\mathbb{F}$ i ciąg $\alpha_1, \ldots, \alpha_n \in \mathbb{F}^n$. Wtedy

$$\mathrm{RS}(\alpha, n, k) = \{(f(\alpha_1), \ldots, f(\alpha_n)) \ : \ f \in \mathbb{F}[X], \deg(f) < k$$

Zauważmy, że to daje naturalna kodowanie kodów RS:

$$\vec{m} \mapsto f_{\vec{m}} = \sum_{i=0}^{k-1} m_i X^i$$

Ale nie jest to jedyne możliwe kodowanie.

Lemat 5.2. *Kod RS jest kodem liniowym.*

Można przez macierze Vandermonde'a, ale można też bezpośrednio, że jest to przestrzeń liniowa.

Lemat 5.3. *Kod Reeda Salomona jest $[n, k, n-k+1]_q$-kodem.*

Dowód. Wystarczy popatrzeć, ile może być maksymalnie zer w słowie kodowym ($k-1$: wielomian jest stopnia najwyżej $k-1$!), czyli pozostałe muszą być niezerowe — $n-k+1$ niezerowych.

Można też wprost, licząc odległości między parami słów kodowych. □

Czyli spełniają ograniczenie Singletona. Ale niestety, q musi rosnąć wraz z liczbą błędów ($q \geq n$)

Macierze generatorów: np. wielomiany X^i dla $i = 0, \ldots, k-1$.

$$\begin{bmatrix} 1 & \alpha_1 & \ldots & \alpha_1^{k-1} \\ 1 & \alpha_2^1 & \ldots & \alpha_2^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_n^1 & \ldots & \alpha_n^{k-1} \end{bmatrix}$$

Z algebry liniowej wiemy, że kwadratowa Macierz Vendermonde'a jest odwracalna. Czyli też każda podmacierz kwadratowa macierzy Vandermonde'a jest odwracalna.

Przykład/Zastosowanie 5.4. Kodowanie CD: podwójny RS: najpierw $(32, 28)$, jako symboli używamy bajtów. Bloki, których nie potrafi odczytać oznacza jako wymazane. Następnie te bajty są rozrzucane i stosowany jest $(28, 24)$ kod. Podobno z dużym zapasem to zrobiono. DVD podobnie, inne wielkości.

Kody dwuwymiarowe: PDF-417, MaxiCode, Datamatrix, QR Code, i Aztec Code. Różnie, również definiowalne (0 do 1/3 bitów kontrolnych)

QR: wężykiem, korekcja na bitach, 1 bit to 2×4 lub 4×2, 4 poziomy korekcji (do 1/3).

Do pewnego momentu: standard jako kod zewnętrzny w transmisji satelitarnej (kod zewnętrzny w stosunku do kodów konwolucyjnych).

5.2 Dekodowanie kodów RS: Berlekamp–Welch

Czas $\mathcal{O}(n^3)$, ale w miarę prosta idea. [8, Rozdział 15] zawiera obszerną intuicję, której kompletnie nie rozumiem.

Z grubsza działa też dla ogólnych, tylko trzeba trochę pokombinować.

Cel: dane $\vec{w} = w_0, \ldots, w_n - 1 \in \mathbb{F}^n$.
szukane: wielomian $f \in \mathbb{F}[X]$, t. że $\deg(f) < k$, $f(\alpha_i) \neq w_i$ dla najwyżej $e \leq \left\lfloor \frac{n-k}{2} \right\rfloor$ albo „?" jak nie ma takiego wielomianu.

Oznaczenie: $I = \{i : f(\alpha_i) \neq w_i\}$. Dobrze zdefiniowane, bo takie f jest jedyne. Jak nie ma wyniku, to trudno.

Moglibyśmy po prostu wybrać te błędy, zinterpolować i rozwiązać...

Popatrzmy na wielomian

Definicja 5.5 (Error locator polynomial)**.** Dla zbioru pozycji błędów I zdefiniujmy *error locator polynomial*:

$$E(X) = \prod_{i \in I} (X - \alpha_i) \ .$$

Dlaczego taki: głównie to wiedziano, że go należy użyć... Zauważmy, że

$$\forall_i \ w_i E(\alpha_i) = f(\alpha_i) Q(\alpha_i)$$

Dowód. Jeśli $i \in I$ to obie strony są równe 0 ze względu na E. Jeśli nie, to $w_i = f(\alpha_i)$ i jest OK. □

f ma $\leq k$ współczynników, E ma $\leq \left\lfloor \frac{n-k}{2} \right\rfloor$ współczynników, czyli mamy n punktów danych i mniej zmiennych. Da się rozwiązać!

Ale to sa równanie kwadratowe — w ogólności to jet NP-trudne.

Chcemy:

BW1 wielomian $E(X)$, o wiodącym współczynniku 1, stopnia $e \leq \left\lfloor \frac{n-k}{2} \right\rfloor$

BW2 wielomian Q stopnia $\leq e + k - 1$

BW3 dla każdego i zachodzi $w_i E(\alpha_i) = Q(\alpha_i)$

BW4 nasz wynik to Q/E (jako wielomian)

Uwaga. Jeśli Q/E nie jest zdefiniowane, bo się dzieli z resztą, albo ma za duży stopień, to zwracamy błąd.

1. Czy to w ogóle działa?

2. Jak to zrobić?

3. Złożoność?

Lemat 5.6. *Jeśli dla danego $\vec{w}$ istnieje $\vec{w'} \in \mathrm{RS}$ takie że $\mathrm{d}(w, w') \leq e$ to istnieją Q, E spełniające BW.*

Dowód.

$$E(X) = \prod_{i \in I}(X - \alpha_i)$$
$$Q(X) = E(X)f(X)$$ □

Lemat 5.7. *Jeśli Q_1, E_1 oraz Q_2, E_2 spełniają BW, to $Q_1/E_1 = Q_2/E_2$.*

Uwaga. Zauważmy, że jest to równość ilorazów i reszt, tzn. może być, że oba dzielenia dają resztę. Ale jeśli jeden się dzieli bez reszty, to drugi też, tj. jeśli jest poprawny wynik algorytmu, to wszystkie zwracane dają to samo.

Dowód. Rozpatrzmy wielomian

$$Q_1E_2 - Q_2E_1$$

To jest wielomian stopnia $\leq 2e + k - 1 < n$. Rozpatrzmy jego wartość w α_i:

$$w_iE_1(\alpha_i)E_2(\alpha_i) - w_iE_2(\alpha_i)E_1(\alpha_i) = 0$$

Czyli wielomian $Q_1E_2 - Q_2E_1$ jest zerem. W takim razie

$$Q_1E_2 = Q_2E_1 \implies \frac{Q_1}{E_1} = \frac{Q_2}{E_2}$$ □

To jak to odtworzyć? Będziemy interpolować wielomiany. Zauważmy, że nie interesuje nas, czy ten układ jest jednoznacznie określony, może być nadokreślony lub podokreslony — dowolne rozwiązanie jest OK i wiemy, że jakieś jest.

Czas działania

Trzeba rozwiązać układ równań: $\mathcal{O}(n^3)$ (da się może ciut szybciej w zależność od danych) oraz podzielić dwa wielomiany — n^3. Ponownie dla specyficznych wartości może być ciut lepiej.

5.3 Macierz parzystości kodów RS

Przypomnienie $\mathbb{F}^*$ to zbiór elementów odwracalnych w $\mathbb{F}$. Jest to grupa cykliczna ze względu na mnożenie w ciele. Innymi słowy, ma generator.

Lemat 5.8. *Dla $0 < d < q - 1$*

$$\sum_{\alpha \in \mathbb{F}_q} \alpha^d = \sum_{\alpha \in \mathbb{F}_q^*} \alpha^d = 0$$

Rachunkowy dowód pozostawimy jako ćwiczenie. Zauważmy, że to nic nie znaczy dla $\mathbb{F}_2$, bo wtedy $0 < d < 1$ i nie ma takiej wartości d. Ale też kody RS nad $\mathbb{F}_2$ nie są interesujące.

Twierdzenie 5.9. *Niech $n = q - 1$, niech γ to generator $\mathbb{F}_q^*$. Wtedy*

$$\mathrm{RS}(\gamma^0, \ldots, \gamma^{n-1}, n, k) = \{(c_0, \ldots, c_{n-1}) \,:\, c(\gamma^i) = 0, i = 1, \ldots, n-k\} \ ,$$

gdzie $c(X) = \sum_{i-0}^{n-1} c_i X^i$.

W szczególności, ich macierz parzystości ma postać:

$$\begin{bmatrix} 1 & \gamma & \gamma^2 & \ldots & \gamma^{n-1} \\ 1 & \gamma^2 & \gamma^4 & \ldots & \gamma^{2(n-1)} \\ \vdots & \vdots & \ldots & \vdots & \\ 1 & \gamma^{n-k} & \gamma^{(n-k)\cdot 2} & \ldots & \gamma^{(n-k)(n-1)} \end{bmatrix}$$

Dowód. Wystarczy pokazać, że iloczyn macierzy generującej i macierzy parzystości jest 0. Czyli wystarczy pokazać, że iloczyn dowolnego wiersza H oraz kolumny G jest 0. Sprawdźmy:

$$\begin{aligned} [1 \ \gamma^i \ \gamma^{2i} \ \ldots \ \gamma^{(n-1)i}] \cdot \begin{bmatrix} 1 \\ \gamma^j \\ \gamma^{2j} \\ \vdots \\ \gamma^{(n-1)j} \end{bmatrix} &= \sum_{\ell=0}^{n-1} \gamma^{\ell(i+j)} \\ &= \sum_{\alpha \in \mathbb{F}} \alpha^{(i+j)} \end{aligned}$$

Zauważmy, że $2 \leq i + j \leq n - k + k - 1 = q - 1$, czyli $i + j$ spełnia założenia Lematu 5.8 i tym samym

$$\sum_{\alpha \in \mathbb{F}} \alpha^{(i+j)} = 0 \qquad \square$$

Sporo skorzystaliśmy z wyboru punktów oraz $n = q - 1$. Ale ogólnie to też jest prawda (ćwiczenie)

Definicja 5.10. Uogólnione kody Reeda Solomona:

$$\mathrm{GRS}(\vec{\alpha}, n, k, \vec{\lambda}) = \{(\lambda_0 f(\alpha_0), \lambda_1 f(\alpha_1) \ldots, \lambda_{n-1} f(\alpha_{n-1}) \,:\, f \in \mathbb{F}[X], \deg(f) < k\}.$$

Twierdzenie 5.11. $\mathrm{GRS}(\vec{\alpha}, n, k, \vec{\lambda})^\perp = \mathrm{GRS}(\vec{\alpha}, n, n-k, \vec{\sigma})$ *dla pewnego* $\vec{\sigma} \in \mathbb{F}^{n-k}$.

Nie takie trudne, znając podejście — jako ćwiczenie.

Rozdział 6

Algorytm Berlekamp-Massey

Na podstawie [9, Rozdział 5] z pewnymi uproszczeniami w rachunkach. Jest też w [6], dokładniej, ale z inną notacją (chyba bardziej standardową).

Czas działania: $\mathcal{O}(n^2)$. Nie jest najlepszy asymptotycznie, ale dobrze się implementuje w hardwarze.

Ważna cecha: czas działania zależy od faktycznej liczby błędów (główna część podprocedury). Jest prosty w implementacji w chipach, warianty tego algorytmu były implementowane sprzętowo.

Będziemy to robić na podstawie macierzy parzystości, z syndromu obliczymy "error locator polynomial".

Będę trochę oszukiwał — pewne uproszczenia...

Przypomnijmy sobie macierz parzystości kodów RS (dla właściwego wyboru punktów):

$$H = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \gamma & \gamma^2 & \cdots & \gamma^{n-1} \\ 1 & \gamma^2 & \gamma^4 & \cdots & \gamma^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \gamma^{n-k-1} & \gamma^{2(n-k-1)} & \cdots & \gamma^{(n-k-1)(n-1)} \end{bmatrix}$$

Tego górnego wiersza tak naprawdę nie ma, ale dopisujemy, żeby było bardziej czytelnie; jest za to dodatkowy wiersza na dole. Wszystkie obliczenia da się zrobić.

Przypomnijmy, że syndrom to

$$Hw = H(v + \vec{e}) = H\vec{e} \ .$$

Od tego momentu $\vec{e}$ to błąd, zaś $t = \|\vec{e}\|_1$ to liczba błędów. Nie znamy, ale pojawi się w oszacowaniach.

Error locator polynomial to

$$E(X) = \prod_{i:e_i \neq 0} (X - \gamma^i) \ .$$

Uwaga: korzystamy z tego, jakie są punkty ewaluacji (bo z tego też wynika, jaka jest macierz parzystości). Inne punkty dają coś podobnego, ale nie byłoby tak prosto liczyć.

Nie mamy $\vec{e}$, ale mamy $H\vec{e}$. Weźmy $\vec{f} = (f_0, \ldots, f_{n-k-1})$.

$$\vec{f} H \vec{e}$$

Zastanówmy się, jak wygląda $\vec{f}H$. Po transpozycji mamy

$$H^T \vec{f}^T$$

Zauważmy, że macierz H jest macierzą generatorów kodu RS (wymiaru $n-k$)! Czyli

$$\vec{f}H = [f(1),\ \ f(\gamma),\ \ \ldots,\ \ f(\gamma^{n-1})]$$

(tak naprawdę to wszystkie te wartości są pomnożone przez γ, da się podzielić — to jest to nasze małe oszustwo).

Zdefiniujmy

$$\langle \vec{e}, f\rangle = \langle \vec{e}, f(X)\rangle = \vec{f}H\vec{e} = \sum_{i=0}^{n-1} e_i f(\gamma^i)$$

Zauważmy, że mamy dostęp do tej funkcji, prawie jak do wyroczni, tzn. potrafimy policzyć w czasie $\mathcal{O}(n-k)$:

$$f \mapsto \langle \vec{e}, f\rangle$$

Używając tej funkcji (jako czarnej skrzynki), odtworzymy $E(X)$. Potem sfaktoryzujemy (można naiwnie, są też lepsze metody) i to nam da gdzie są błędy. Wtedy łatwo — na pozostałych punktach można zinterpolować. Ważne: możemy policzyć tę funkcję dla każdego wielomianu stopnia $\leq n-k-1$.

Najpierw prosta obserwacja:

$$\langle \vec{e}, E\rangle = \sum_{i=0}^{n-1} \underbrace{e_i E(\gamma^i)}_{\text{jedno lub drugie} = 0} = 0$$

Co więcej, to jest też prawda, jak przemnożymy E przez dowolny wielomian

Lemat 6.1.

$$\forall\, \langle \vec{e}, gE\rangle = 0$$

Dowód.

$$\langle \vec{e}, Eg\rangle = \sum_{i=0}^{n-1} \underbrace{e_i E(\gamma^i)}_{\text{jedno lub drugie} = 0} g(\gamma^{im}) \quad \square$$

Cel: postaramy się skonstruować f małego stopnia, taki że dla każdego wielomianu

$$\langle \vec{e}, gf\rangle = 0$$

Trzeba pokazać, że jest dobry, no i jak go skonstruować.

Uwaga. Uwaga: mamy dostęp do tej funkcji tylko dla wielomianów stopnia mniejszegos niż $n-k$, trzeba sprawdzić, że nie próbujemy policzyć większych!

Najpierw, że to naprawdę daje rozwiązanie:

Lemat 6.2. *Niech* $\|\vec{e}\|_1 = t$, *oraz* $\forall_{0\leq r<t} \langle \vec{e}, X^r f\rangle = 0$. *Wtedy* $E(X)|f(x)$, *gdzie* $E(X) = \prod_{i:e_i\neq 0}(X-\gamma^i)$.

W szczególności, jeśli $\deg(f) \leq t$ *to* $f(X) = \alpha E(X)$ *dla pewnego* $\alpha \in \mathbb{F}$.

Dowód. Z liniowości mamy, że $\deg(g) < t$ implikuje

$$\langle \vec{e}, gf\rangle = 0$$

Zdefiniujmy $E^{(k)} = E/(X-\gamma^k)$, gdzie $e_k \neq 0$, tj. $E^{(k)}$ jest wielomianem (stopnia $t-1$). Wtedy

$$\begin{aligned} 0 &= \left\langle \vec{e}, E^{(k)} f \right\rangle \\ &= \sum_{i=0}^{n-1} \underbrace{e_i E^{(k)}(\gamma_i)}_{=0 \text{ dla } i\neq k} f(\gamma^i) \\ &= \underbrace{e_k}_{\neq 0} \underbrace{E^{(k)}(\gamma_k)}_{\neq 0} f(\gamma^k) \end{aligned}$$

Czyli $(X - \gamma^k)|f$ dla każdego k takiego że $e_k = 1$. Czyli też $E|f$.

Jako że $\deg(E) = t$ to $\deg(f) \leq t$ implikuje $f = \alpha E$. □

Wniosek 6.3. Niech $\|\vec{e}\|_1 = t$. Jeśli $\langle \vec{e}, X^r f \rangle = 0$ dla $r = 0, 1, \ldots, t-1$ to $\langle \vec{e}, X^r f \rangle = 0$ dla każdego r.

Dowód. Z Lematu 6.2 mamy, że $E|f$, czyli $f = f'E$ Ale w takim razie z Lematu 6.1

$$\langle \vec{e}, X^r f \rangle = \langle \vec{e}, X^r f' E \rangle = 0$$ □

Definicja 6.4.

$$\begin{aligned} \operatorname{span}(f) &= \min\{r \,:\, \langle \vec{e}, X^r f \rangle \neq 0\} \;; \\ \operatorname{disc}(f) &= \left\langle \vec{e}, X^{\operatorname{span}(f)} f \right\rangle \;. \end{aligned}$$

Wniosek 6.5. Niech $\|\vec{e}\|_1 = t$. Jeśli $\operatorname{span}(f) \geq t$ to $\operatorname{span}(f) = +\infty$.

Przeformułowanie poprzedniego wniosku.

Najważniejsza konsekwencja: wystarczy, że policzymy do spanu t i koniec.

Będziemy iteracyjnie konstruować f o coraz większym span, w końcu musimy trafić na ∞. Szczegóły algorytmu dadzą, że to jest E.

Jak z dwóch o mniejszym spanie zrobić jeden o większym?

Lemat 6.6. *Niech $f \neq f'$: dwa różne wielomiany,* $\operatorname{span}(f) = r < \infty$, $\operatorname{disc}(f) = \mu$ *oraz* $\operatorname{span}(f') = r' < \infty$, $\operatorname{disc}(f') = \mu'$, *bzo* $r' \geq r$.

Zdefiniujmy $h(X) = f(x) - \frac{\mu}{\mu'} X^{r'-r} f'$. *Wtedy* $\operatorname{span}(h) > \operatorname{span}(f)$.

Uwaga. Zauważmy, że to podwyższa *mniejszy* span. Ale to jest OK, bo jak sie zrównają, to i tak podwyższymy.

Dowód. Rozważmy

$$\left\langle \vec{e}, X^i h \right\rangle = \left\langle \vec{e}, X^i f \right\rangle - \frac{\mu}{\mu'} \left\langle \vec{e}, X^{r'-r+i} f' \right\rangle$$

Dla $i = r$ dostajemy

$$\begin{aligned} \langle \vec{e}, X^r h \rangle &= \langle \vec{e}, X^r f \rangle - \frac{\mu}{\mu'} \left\langle \vec{e}, X^{r'} f' \right\rangle \\ &= \mu - \frac{\mu}{\mu'} \mu' = 0 \end{aligned}$$

Dla $i < r$ wychodzi 0, bo span kombinacji liniowej to przynajmniej minimum spanów. (Można też policzyć wprost). □

Wykorzystamy to do iteracyjnego algorytmu, który trzyma dwa wielomiany.

Uwaga. Ten algorytm ma zaszytą inicjalizację: analiza itp. jest inna dla $g = 0$, po zmianie na $g \neq 0$ już nigdy nie wracamy do $g = 0$. Warto to zanalizować osobno. W szczególności *korzystamy* tam z założenia, że $\deg(0) < 0$.

Lemat 6.7 (Niezmienniki)**.** *Po m-tej iteracji mamy:*

- *f ma wiodący współczynnik* 1
- $\deg(f) + \operatorname{span}(f) > m$
- $g = 0$ *lub*

Algorytm 1 Algorytm Berlekamp-Messey

$f \leftarrow 1, g \leftarrow 0$
for $m \leftarrow 0, \ldots, 2t-1$ **do** ▷ t nie jest potrzebne, wystarczy górne ograniczenie
  $c \leftarrow \deg(f) - 1$
  $r \leftarrow m - c - 1$ ▷ $r = m - \deg(f)$
  ▷ Niezmienniki: $\operatorname{span}(f) \geq r$, $\operatorname{span}(g) = c$, $\operatorname{disc}(g) = 1$
  $\mu \leftarrow \langle \vec{e}, X^r f \rangle$ ▷ Da się dla $m \leq 2t - 1 \leq n - k - 1$
  if $\mu \neq 0$ **then** ▷ Jeśli $\mu = 0$ to f, g nie zmieniają się
    ▷ Tutaj mamy $\operatorname{span}(f) = r$.
    if $r \leq c$ **then**
      $f' \leftarrow f(X) - \mu X^{c-r} g(X)$ ▷ Czyli update jak z Lematu 6.6
      $g' \leftarrow g(X)$
    else
      $f' \leftarrow X^{r-c} f(X) - \mu g(X)$ ▷ Czyli update jak z Lematu 6.6 razy $-\mu^{-1}$
      $g' \leftarrow \mu^{-1} f(X)$
    $f, g \leftarrow f', g'$
return f

- $\operatorname{span}(g) = \deg(f) - 1$
- $\deg(g) + \operatorname{span}(g) \leq m$
- $\operatorname{disc}(g) = 1$

Dowód. Przez indukcję. Na początku zachodzi ($m = -1$).

Warto jest zanalizować osobno przypadek, gdy $g = 0$, pozostawimy to jako ćwiczenie.

$\mu = 0$ W tym przypadku algorytm nic nie zmienia. Wszystkie niezmienniki trywialnie zachodzą: jedyny wymagający sprawdzenia to $\deg(f) + \operatorname{span}(f) \geq m$, bo m się zwiększyło, ale sprawdziliśmy właśnie, że jest OK.

$\mu \neq 0$ W przeciwnym przypadku mamy, że $r = \operatorname{span}(f)$: z założenia indukcyjnego mamy, że $\operatorname{span}(f) + \deg(f) > m - 1$ a teraz sprawdziliśmy, że $\operatorname{span}(f) + \deg(f) \leq m$, czyli jest równość. Z założenia indukcyjnego $\operatorname{span}(g) = \deg(f) - 1 = c$ oraz $\operatorname{disc}(g) = 1$

$r \leq c$ Podstawienie jest dokładnie takie, jak mówi Lemat 6.6, w szczególności $\operatorname{span}(f') > r$.

Zauważmy, że $\deg(f) = \deg(f')$:

$$\begin{aligned} \deg(g) &\leq (m-1) - \operatorname{span}(g) && \text{założenie indukcyjne} \\ &= m - 1 - c && \text{założenie indukcyjne} \end{aligned}$$

Czyli w updacie „$-\mu X^{c-r} g(X)$”:

$$\begin{aligned} \deg(-\mu X^{c-r} g(X)) &\leq (c-r) + (m-1-c) && (6.1) \\ &= m - 1 - r \\ &= m - 1 - (m - \deg(f)) && \text{z określenia } r \\ &= \deg(f) - 1 \end{aligned}$$

I tym samym

$$\deg(f') = \deg(f)$$

Co daje

$$\begin{aligned}\deg(f') + \operatorname{span}(f') &> \deg(f) + \operatorname{span}(f) \\ &> m - 1\end{aligned}$$

Co daje

$$\deg(f') + \operatorname{span}(f') > m$$

Zauważmy, że (6.1) daje też, że f' ma wyraz wiodący taki, jak f.

Jako że g się nie zmienia, to warunki na nie się zachowują (znowu korzystamy z tego, że stopień f się zachowuje).

$r > c$ Zauważmy, update z Lematu daje wielomian o większym spanie równy

$$g(x) - \frac{1}{\mu} X^{r-c} f$$

Jak się to przemnoży przez $-\mu$ to otrzymujemy wzór na f'. Reszta dowodu jak powyżej, jako ćwiczenie. □

Lemat 6.8. *Jeśli* $\|\vec{e}\|_1 = t$ *to* $f(X) = E(X)$ *dla* $m = 2t - 1$.

Dowód. Zauważmy najpierw, że

$$\deg(f(X)) \leq t \ .$$

Łatwo sprawdzić, że $g = 0$ oznacza, że $f = 1$ i teza w oczywisty sposób zachodzi. A jeśli $g' \neq 0$ to $\operatorname{span}(g') = \deg(f') - 1 \geq t$. Ale wiemy z Wniosku 6.5, że $\operatorname{span}(g') \geq t$ oznacza, że $\operatorname{span}(g) = \infty$, co nie jest możliwe.

Teraz

$$\deg(f) + \operatorname{span}(f) > m$$

daje

$$\begin{aligned}\operatorname{span}(f) &> m - \deg(f) \\ &\geq 2t - 1 - t \\ &= t - 1\end{aligned}$$

Czyli $\operatorname{span}(f) \geq t$ i tym samym $\operatorname{span}(f) = \infty$. Czyli $E|f$, ale $\deg(f) \leq t = \deg(E)$, czyli $E = \alpha f$, ale f ma wiodący współczynnik 1, tak jak E, czyli $E = f$. □

Mamy już error locator polynomial. Teraz trzeba go sfaktoryzować. Można to zrobić naiwnie w czasie $\mathcal{O}(tn)$: podstawiać kolejne liczby z $\mathbb{F}_q$ i liczyć wartość, wiadomo, jak wygląda, to nie trzeba więcej). Da się też lepiej.

Następnie znając błędy możemy zinterpolować wielomian na poprawnych współczynnikach. Uwaga, mamy sporą nadwyżkę (potrzebujemy k, mamy $k + \frac{n-k}{2}$). Liczenie tego naiwnie zajmuje czas $\mathcal{O}(k^3)$. Ale można inaczej — pokażemy.

Rozdział 7

Zastosowanie: group testing

Głównie [9, Wykład 8], opisane też w [8, Rozdział 19], w tym drugim moim zdaniem trochę przegadane i przeformalizowane.

Problem: mamy populację N osób, z których $\leq d$ jest chorych. Możemy pobrać próbki, zmieszać i przetestować. Test wychodzi pozytywny, jeśli choć jedna jest chora. Celem jest zidentyfikowanie wszystkich chorych przy jak najmniejszej liczbie testów.

Takie mieszanie ma sens, jeśli chorych jest istotnie mniej, niż osób (jeśli $d = N/2$ to nie wiele można zrobić)

Dwa modele: adaptacyjny i nieadaptacyjny: czy możemy zmieniać testy w zależności od wyników poprzednich? Oba mają swoje uzasadnienie.

7.1 Formalizacja

- N: liczba osób, wektor $\vec{x} = (x_1, \ldots, x_N)^T \in \{0,1\}^N$
- d: ograniczenie górne na liczbę zarażonych: $\|x\|_1 \leq d$
- pojedynczy test $\vec{t}$ to również wektor z $\{0,1\}^N$, ale nie sumujemy, tylko bierzemy alternatywę, tj. wynik to $\bigvee_{i=1}^{N} t_i \cdot x_i$
- chcemy na podstawie wyników jednoznacznie odtworzyć $\vec{x}$.
- model adaptacyjny
- model nieadaptacyjny (ślepy): testy to macierze $t \times N$, operacje wykonujemy w pierścieniu boolowskim $\{0,1\}$ z operacją koniunkcji oraz alternatywy

Definicja 7.1. $t(d,N)$ to minimalna liczba testów nieadaptacyjnych, które należy wykonać, aby wyznaczyć N.

$t^a(d,N)$ to analogiczna liczba testów adaptacyjnyy.

Chcemy poznać asymptotykę na te liczby.

Fakt 7.2. $t^a(d,N) \leq t(d,N)$

7.2 Wariant adaptacyjny

Lemat 7.3. $t^a(d,N) \geq d \log \frac{N}{d}$

Dowód. Zauważmy, że taki algorytm to drzewo decyzyjne (choć niekoniecznie w drugą stronę), zaś głębokość tego drzewa jest ograniczona z góry przez $t^a(d, N)$.

Potrafimy odróżnić każde dwa wejścia, których jest

$$\mathrm{Vol}_2(d, N)$$

czyli musi być:

$$\begin{aligned} t^a(d, N) &\geq \log \mathrm{Vol}_2(d, N) \\ &> \log_2 \binom{N}{d} \\ &> \log_2 \left(\frac{N}{d}\right)^d \\ &= d \log_2 \frac{N}{d} \end{aligned}$$

□

Lemat 7.4. $t^a(d, N) = \mathcal{O}\left(d \log_2 \frac{N}{d}\right)$.

Prosty algorytm pozostawiamy jako ćwiczenie. Czyli w zasadzie można walczyć o stałą.

7.3 Wariant nieadaptacyjny

Reprezentacja algebraiczna: testy to macierze $m \times N$, tylko że mnożenie wykonujemy nad pierścieniem boolowskim $\{0, 1\}$ (tj. alternatywa i konkatenacja zamiast dodawania i mnożenia)

Uwaga. Problem nieadaptacyjny jest podobny do problemu dekodowania z syndromu, *ale* jest nad inną strukturą algebraiczną: nad pierścieniem boolowksim, a nie nad ciałem.

Wprawka: co jeśli $d = 1$? Jeśli nie ma chorego, to wyjdzie 0. Jesli jest chory, to wynik dla i-tego chorego to i-ta kolumna macierzy testów M (i to nie może być 0, bo to wynik dla 0 chorych) Czyli kolumny muszą być różne. To jest kod o odległości 3. Kody Hamminga! (Mały problem ze złym wymiarem, ale nie szkodzi — jako kolumny bierzemy liczby od 1 do N zapisane binarnie).

Wniosek 7.5.

$$t(1, N) \leq \lfloor \log N \rfloor + 1$$

Przeformułujmy dla macierzy, co chcemy

Definicja 7.6. Macierz M rozmiaru $t \times N$ jest d-separowalna, jeśli dla $J, J' \subseteq \{1, \ldots, N\}$, $|J|, |J'| \leq d$ zachodzi

$$\bigvee_{j \in J} M^j \neq \bigvee_{j \in J'} M^j,$$

gdzie M^j to j-ta kolumna M zaś $\bigvee$ to alternatywa logiczna po współrzędnych.

Fakt 7.7. *Problem testowania nieadaptacyjnego w m testach to problem znalezienia d-separowalnych macierzy rozmiaru $m \times N$.*

Dekodowanie naturalne, ale chcielibyśmy to zrobić efektywniej.

Silniejsza definicja, która pozwoli na algorytm (o czasie działania $\mathcal{O}(tN)$)

Definicja 7.8. Macierz M rozmiaru $t \times N$ jest *d-rozłączna* wtedy i tylko wtedy, gdy dla każdego $S \subseteq \{1, \ldots, N\}$, $|S| \leq d$ oraz dla każdego $j \notin S$, istnieje $i \in \{1, \ldots, t\}$ takie, że $M_{ij} = 1$ oraz dla wszystkich $k \in S$, $M_{ik} = 0$. Równoważnie

$$M^j \not\subseteq \bigvee_{k \in S} M^k$$

Ta formalna definicja ma sens: jeśli S to zbiór chorych, to i-ty test daje dowód, że nic poza S nie jest chore. Tzn. mamy sposób na określenie „negatywnych"

Lemat 7.9. *Jeśli M jest d-rozłączna to jest też d-separowalna.*

Lemat 7.10. *Jeśli macierz jest d-rozłączna, to w czasie $\mathcal{O}(tN)$ odtworzyć zbiór $\leq t$ jedynek z wejścia.*

Dowód pozostawimy jako proste ćwiczenie.

7.4 Konstrukcja macierzy d-rozłącznych

Kautz-Singleton '64.

Idea: kolumny mają być „daleko" — użyjemy słów kodowych.

N — liczba słów kodowych. Ustawiamy je wszystkie obok siebie, następnie zamieniamy $i \in \mathbb{F}_q$ na wektor $e_i \in \mathbb{F}_2^i$. W ten sposób $m = nq$, gdzie n to długość słowa kodowego.

Bierzemy $\mathrm{RS}_q(n,k)$, potem ustalimy parametry.

Twierdzenie 7.11. *Jeśli $\mathrm{d}(C) > n\frac{d-1}{d} = n - \frac{n}{d}$ to ta konstrukcja daje d-rozłączną macierz.*

Dowód. Weźmy zbiór kolumn S. Chcemy pokazać, że każde inne słowo kodowe ma na jakiejś pozycji coś innego, niż dowolne z S.

Przypomnijmy, że $\mathrm{d}(C) = n-k+1$. Zauważmy, że dwa dowolne słowa kodowe mają $k-1 = n - \mathrm{d}(C)$ wspólnych pozycji. Czyli suma takich pozycji to

$$\begin{aligned} d \cdot (n - \mathrm{d}(C)) &< d\left(n - \left(n - \frac{n}{d}\right)\right) \\ &= n \end{aligned}$$

Czyli na jakiejś pozycji jest inne! □

Jeśli zaczniemy od $\mathrm{RS}_q(\mathbb{F}_q, q, k)$, to $n = q$, $\mathrm{d}(C) = q - k + 1$. Weźmy

$$k = \left\lceil \frac{q}{d} \right\rceil \ .$$

Wtedy

$$\begin{aligned} \mathrm{d}(C) &= n - k + 1 \\ &= q - \left\lceil \frac{q}{d} \right\rceil + 1 \\ &= q - \left(\left\lceil \frac{q}{d} \right\rceil - 1 \right) \\ &> q - \frac{q}{d} \ . \end{aligned}$$

Czyli macierze ma $nq = q^2$ na $q^k = q^{\lceil q/d \rceil}$. Co daje $\log_q(N) = \left\lceil \frac{\sqrt{m}}{d} \right\rceil$, czyli

$$\sqrt{m} \approx d \log_q(N) = \frac{d \log(N)}{\frac{1}{2}\log m}$$

Czyli

$$\sqrt{m} \log m = \mathcal{O}\left(d \log(N)\right)$$

Proste rachunki pokazują, że to daje

$$\sqrt{m} = \mathcal{O}\left(\frac{d \log N}{\log(d \log N)}\right)$$

Czyli

$$m = \mathcal{O}\left(\frac{d^2 \log^2 N}{(\log \log N)^2 + \log^2 d}\right)$$

Ta konstrukcję da się sparametryzować, tj. podać wartości w zależności od parametru kodu.

Rozdział 8

Obliczanie błędów z error locator polynomial (Algorytm Forney'a).

Na podstawie [6].

Najpierw dwie dygresje:

Lemat 8.1. *W ciele $\mathbb{F}_q$ i $\beta \in \mathbb{F}_q$ zachodzi:*

$$\frac{x^n - 1}{x - \beta} = \sum_{i=0}^{n-1} \beta^i x^{n-1-i}$$

Prosty dowód pozostawimy jako ćwiczenie.

Będzie trochę inna notacja, ale ona jest chyba standardowa... Dla przypomnienia, pracujemy z kodami $\mathrm{RS}((\gamma^0, \gamma^1, \ldots, \gamma^{k-1}), q-1, k)$. Przekształcenie

$$f \mapsto \sum_{i=0}^{n-1} e_i f(\gamma_i)$$

będziemy oznaczać przez $S^e(f)$ (w starej notacji: $\langle \vec{e}, f \rangle$) Ponownie, dla przypomnienia, $S^e(f)$ można obliczyć mnożąc wektor f^T przez syndrom (z prawej strony) dla f odpowiednio niskiego stopnia ($\deg(f) < n - k$).

Definicja 8.2. Dla wektora błędów $\vec{e}$ zdefiniujmy $s_m = S^{\vec{e}}(x^m)$.

Zauważmy, że $s_0, \ldots, s_{n-k-1}$ to są kolejne elementy syndromu (to jest wektora $H_C w\vec{e}$, gdzie $\vec{e}$ jest wektorem błędów).

Zauważmy, że wiedząc, gdzie są błędy, możemy je prosto policzyć w czasie $\mathcal{O}(n^3)$, bo

$$s_i = \sum_j e_j \alpha_j^i$$

Gdy wiemy, gdzie są niezerowe e_j to dostajemy układ $n - k$ równań (syndromy) na $\leq \frac{n-k}{2}$ zmiennych (jest spora nadwyżka).

Ale będziemy chcieli to policzyć w czasie $\mathcal{O}(n^2)$.

Definicja 8.3. *Wielomian syndromu* to

$$s^{\vec{e}}(x) = \sum_{i=0}^{n-1} s_i x^{n-1-i}$$

Należy o tym myśleć trochę w terminach funkcji tworzącej.

Lemat 8.4.

$$s^{\vec{e}}(x) = \sum_{i=0}^{n-1} e_i \frac{x^n - 1}{x - \gamma^i}$$

Proste przeliczenie z zależności na początku

$$\begin{aligned}
\sum_{i=0}^{n-1} e_i \frac{x^n - 1}{x - \gamma^i} &= \sum_{i=0}^{n-1} e_i \sum_{m=0}^{n-1} x^{n-1-m} \gamma^{im} \\
&= \sum_{m=0}^{n-1} x^{n-1-m} \underbrace{\sum_{i=0}^{n-1} e_i \alpha^{im}}_{=s_m} \\
&= \sum_{m=0}^{n-1} x^{n-1-m} s_m \\
&= s^{\vec{e}}(x)
\end{aligned}$$

Zauważmy, że $s^{\vec{e}}$ może być użyty do obliczenia wartości błędów:

$$s^{\vec{e}}(\gamma^j) = \sum_{i=0}^{n-1} e_i \frac{(\gamma^j)^n - 1}{\gamma^j - \gamma^i}$$

W tej sumie to się nie zeruje tylko dla $i = j$ i dla niej korzystamy z Lematu 8.1

$$\begin{aligned}
&= e_j \frac{(\gamma^j)^n - 1}{\gamma^j - \gamma^j} \\
&= e_j \sum_{i=0}^{n-1} (\gamma^j)^i (\gamma^j)^{n-1-i} \\
&= e_j n (\gamma^j)^{n-1}
\end{aligned}$$

Jedyny problem: nie znamy współczynników wielomianu $s^{\vec{e}}$. Tak naprawdę to można je policzyć (zadanie), ale można też licząc mniej. Zauważmy, że $s^{\vec{e}}$ zeruje się dla każdego γ^j, takiego, że $e_j = 0$. Ale to oznacza, że po pomnożeniu przez $E^{\vec{e}}$ otrzymamy wielomian, który zeruje się w każdym punkcie, czyli dzieli się przez $x^n - 1$. Formalnie:

Wielomian syndromów może być użyty do scharakteryzowania error locator polynomial.

Lemat 8.5. *Niech $s^{\vec{e}}(X)$ będzie wielomianem syndromu dla $\vec{e}$ i niech $f(x)$ będzie dowolnym wielomianem. Wtedy $f(x)s^{\vec{e}}(x)$ dzieli się przez $x^n - 1$ wtedy i tylko wtedy, gdy $E(x)$ dzieli $f(x)$.*

Dowód. $\Leftarrow$ Wystarczy pokazać dla $f = E$.

$$\begin{aligned}
E(x)s^{\vec{e}}(x) &= \left(\prod_{j:e_j \neq 0} (x - \gamma^j) \right) (x^n - 1) \sum_{i=0}^{n-1} \frac{e_i}{x - \gamma^i} \\
&= \left(\prod_{j:e_j \neq 0} (x - \gamma^j) \right) (x^n - 1) \sum_{i:e_i \neq 0} \frac{e_i}{x - \gamma^i} \\
&= (x^n - 1) \sum_{i:e_i \neq 0} e_i \prod_{\substack{j:e_j \neq 0 \\ j \neq i}} (x - \gamma^j)
\end{aligned}$$

Czyli $x^n - 1$ dzieli $E(x)s^{\vec{e}}(x)$.

$\Rightarrow$ Załóżmy, że $(x^n - 1) | f(x)s^{\vec{e}}(x)$. Co jest równoważne temu, że każde $x - \gamma^i | f(x)s^{\vec{e}}(x)$. Przypomnijmy, że

$$s^{\vec{e}}(x) = \sum_{j=0}^{n-1} e_j \frac{x^n - 1}{x - \gamma^j} \ .$$

W tej sumie tylko jeden wyraz nie dzieli się przez $x - \gamma^i$, ten dla $j = i$. I on naprawdę się nie dzieli. Czyli $x - \gamma^i | f(x)$, dla każdego i t. że $e_i \neq 0$. Czyli też $E|f$. □

Definicja 8.6. Niech $s^{\vec{e}}(x)$ będzie wielomianem syndromu. Niech $\varphi^{\vec{e}}(x)$ będzie wielomianem takim że

$$E^{\vec{e}}(x)s^{\vec{e}}(x) = \varphi^{\vec{e}}(x)(x^n - 1)$$

Wtedy $\varphi^{\vec{e}}(x)$ to wielomian ewaluacji błędów dla $\vec{e}$.

Powodem, dla którego nazywamy $\varphi^{\vec{e}}(x)$ ewaluatorem błędów $\vec{e}$ jest to, że może on być użyty do obliczenia wartości błędów. Będziemy do tego potrzebować pochodnych.

Pojawia się pytanie, czy możemy policzyć na podstawie tej definicji $\varphi^{\vec{e}}$? Nie można tego policzyć wprost, tj. pomnożyć $E^{\vec{e}}$ oraz $s^{\vec{e}}$ i potem podzielić przez $x^n - 1$, bo nie potrafimy policzyć s_m dla dużych m — teoretycznie są zdefiniowane do $n - 1$, potrafimy policzyć do $n - k - 1$ (dla przypomnienia, obliczamy je z syndromu). Ale jak się policzy dokładniej to okazuje się, że nie będziemy ich potrzebować.

Lemat 8.7. *Wielomian $\varphi^{\vec{e}}$ można policzyć znając E oraz $s_0, \ldots, s_{n-k-1}$.*

Dowód pozostawimy jako proste ćwiczenie.

Teraz mamy E, φ. Potrafimy już wyliczyć $s^{\vec{e}}$ i policzyć każde e_i. Ale znów, można szybciej: chcemy $s^{\vec{e}}$ tylko po to, by je obliczyć w punkcie γ^i. Ale nie musimy w tym celu wyliczać samego $s^{\vec{e}}$, wystarczy nam zależność $E^{\vec{e}}s^{\vec{e}} = \varphi^{\vec{e}}(x^n - 1)$, bo poza $s^{\vec{e}}$ to wszystko już znamy i potrafimy policzyć w ustalonym punkcie. Ale w γ^i obie strony się zerują (a tak naprawdę to się dzielą w każdym punkcie). Ale to jest pojedynczy pierwiastek, można teraz skorzystać z prostego triku znanego z interpolacji:

Lemat 8.8. *Jeśli α jest jednokrotnym pierwiastkiem wielomianu f to*

$$\left(\frac{f}{x - \alpha}\right)(\alpha) = f'(\alpha)$$

Ta własność jest np. używana przy obliczaniu współczynników interpolacyjnych poprzez wielomiany Lagrange'a.

Dowód. Niech

$$f = (x - \alpha)g \ .$$

Chcemy pokazać, że $g(\alpha) = f'(\alpha)$.

$$\begin{aligned} f'(\alpha) &= ((x - \alpha)g)'(\alpha) \\ &= g(\alpha) + (\alpha - \alpha)g'(\alpha) \\ &= g(\alpha) \end{aligned}$$

□

Po podstawieniu w naszym przypadku i obliczeniach mamy:

Lemat 8.9. *Niech $E(x)$ i $\varphi^{\vec{e}}(x)$ będą error locator polynomial i wielomianem ewaluacji błędów dla $\vec{e}$. Wtedy dla każdego k takiego że $e_k \neq 0$ zachodzi*

$$e_k = \frac{\varphi^{\vec{e}}(\gamma^k)}{E'(\gamma^k)} \ .$$

Dowód pozostawimy jako ćwiczenie.

Pozostaje pytanie, jak to policzyć. Jako że znamy $E^{\vec{e}}$, to można policzyć wprost wyrażenie na pochodną (czas liniowy) i obliczyć w danym punkcie (czas liniowy). Podobnie, skoro znamy $\varphi^{\vec{e}}$, to możemy w czasie liniowym policzyć jego wartość.

Zauważmy, że gdy znamy wartości błędów, czyli cały wektor $\vec{e}$, to możemy go odjąć od komunikatu i dostać upragnione słowo kodowe.

Uwaga. Można rozszerzyć algorytm BM tak, aby liczył też ewaluator błędów. A nawet usunąć go całkowicie i liczyć z prostszych rzeczy. Ale tego już nie pokażę.

Rozdział 9

Algorytm Peterson–Gorenstein–Zierler (baza do Berlekamp-Massey)

Skorzystajmy z definicji z poprzedniej części, tj.:

$$s_j = \sum_{i=0}^{n-1} e_i(\gamma^j)^i$$

Dla $0 \leq j < n-k$ to są elementy syndromu.

Niech error locator polynomial będzie postaci

$$E^{\vec{e}} = \sum_{i=0}^{t} \lambda_j x^j$$

przy czym $\lambda_t = 1$. Wiemy już, że

$$S^{\vec{e}}(x^\ell E^{\vec{e}}) = 0$$

Ale to jest przekształcenie liniowe i tak naprawdę to daje

$$\begin{aligned} 0 &= S^{\vec{e}}(\sum_{i=0}^{t} \lambda_j x^{j+\ell}) \\ &= \sum_{i=0}^{t} \lambda_j S^{\vec{e}}(x^{j+\ell}) \\ &= \sum_{i=0}^{t} \lambda_j s_{j+\ell} \end{aligned}$$

Czyli syndromy spełniają zależność liniową zdefiniowaną przez error locator polynomial. Zwykle przekształca się ją do:

$$-s_{\ell+t} = \sum_{i=0}^{t-1} \lambda_j s_{j+\ell}$$

Trzeba tylko ją policzyć, tzn. wyznaczyć odpowiednie wartości λ no i liczbę błędów t.

Rozpatrzmy układ równań próbujący wyznaczyć takie λ dla danego m:

$$\begin{bmatrix} s_0 & s_1 & \cdots & s_{m-1} \\ s_1 & s_2 & \cdots & s_m \\ \vdots & \vdots & \ddots & \vdots \\ s_{m-1} & s_m & \cdots & s_{2m-2} \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_{m-1} \end{bmatrix} = - \begin{bmatrix} s_m \\ s_{m+1} \\ \vdots \\ s_{2m-1} \end{bmatrix}$$

Zauważmy, że dla $m = t-1$ ten układ ma rozwiązanie (prawdziwe λ). Można pokazać, że:

- to rozwiązanie jest jedyne dla $m = t - 1$
- dla $m > t - 1$ jest więcej niż jedno rozwiązanie

oba fakty pozostawiamy jako proste ćwiczenie (de facto były pokazane przy Algorytmie BS).

Nasz algorytm jest więc następujący: rozpatruje $m = \frac{n-k}{2}, \ldots, 1$ i patrzy, czy odpowiednia macierz jest odwracalna. Jeśli tak, to wylicza $\lambda_0, \ldots, \lambda_{m-1}$ i sprawdza, czy spełniają zależność rekurencyjną dla wszystkich syndromów. Jesli nie, to zmniejsza m o 1.

Czas działania: naiwna implementacja ma czas $\mathcal{O}(n^4)$, można poprawić do $\mathcal{O}(n^3)$ (ćwiczenie).

Algorytm BM liczy z grubsza to samo, tylko konstruuje rozwiązanie od dołu.

Rozdział 10

Kody Reeda-Mullera

Na podstawie [8, Rozdziały 9, 14]

Chcemy zbinaryzować kody RS. Jedno z podejść — patrzymy na wiele zmiennych.

Definicja 10.1. Rozpatrzmy zmienne $X_1, \ldots, X_m$. Przez $(\vec{X})^{\vec{d}}$ rozumiemy $\prod_i X_i^{d_i}$. *Sumaryczny stopień* takiego wielomianu to $\deg(\vec{X}^{\vec{d}}) = \sum_i d_i$. Uogólnia się to na sumy:

$$\deg\left(\sum_{\vec{d}} \alpha_{\vec{d}} \vec{X}^{\vec{d}}\right) = \max\{\deg(\vec{X}^{\vec{d}}) \ : \ \alpha_{\vec{d}} \neq 0\}.$$

Patrzymy też na stopień pojedynczej zmiennej:

$$\deg_{X_i}\left(\vec{X}^{\vec{d}}\right) = d_i$$
$$\deg_{X_i}\left(\sum_{\vec{d}} \alpha_{\vec{d}} \vec{X}^{\vec{d}}\right) = \max\{\deg_{X_i}(\vec{X}^{\vec{d}}) \ : \ \alpha_{\vec{d}} \neq 0\}$$

Kody Reeda-Mullera mają trzy parametry: q—rozmiar ciała, m—liczba zmiennych oraz r—maksymalny sumaryczny stopień.

Definicja 10.2 (Kody Reeda-Mullera)**.** Kod Reeda-Mullera z parametrami q, m, r, oznaczany jako $\mathrm{RM}(q, m, r)$, to zbiór ewaluacji (wartości) dla wszystkich wielomianów wielu zmiennych $\mathbb{F}_q[X_1, \ldots, X_m]$ o sumarycznym stopniu co najwyżej r i stopniu każdej zmiennej najwyżej $q-1$ nad wszystkimi punktami z $\mathbb{F}_q^m$.

Formalnie,

$$\mathrm{RM}(q, m, r) = \{f(\vec{v}) \ : \ f \in \mathbb{F}_q[X_1, \ldots, X_m], \deg(f) \leq r, \forall i \deg_{X_i}(f) \leq q-1, \vec{v} \in \mathbb{F}_q^m\}$$

Na przykład, dla $m = q = 2$ i $r = 1$. Wtedy wszystkie wielomiany dwóch zmiennych o współczynnikach z $\mathbb{F}_2$, stopniu najwyżej 1 to 0, 1, X_1, X_2, $1 + X_1$, $1 + X_2$, $X_1 + X_2$ oraz $1 + X_1 + X_2$. Punkty ewaluacji ustawmy w kolejności jako $(0,0), (0,1), (1,0), (1,1)$. W takim razie

$$\mathrm{RM}(2,2,1) = \{(0,0,0,0), (1,1,1,1), (0,0,1,1), (0,1,0,1), (1,1,0,0), (1,0,1,0), (0,1,1,0), (1,0,0,1)\} \ .$$

Widać, że kod jest liniowy.

Zadaniem jest teraz ocenienie wymiaru (nieoczywiste w ogólności) oraz odległości tego kodu.

Długość słowa kodowego to oczywiście q^m i alfabet: $\mathbb{F}_q$. Odległość i wymiar nie jest w ogólności prosto policzyć.

10.1 Niski stopień

Rozważamy teraz $\mathrm{RM}(q,m,r)$ dla $r<q$, co oznacza, że dodatkowe założenie, że $\deg_{X_i}\le q-1$ jest trywialnie spełnione.

Lemat 10.3. *Jeśli $r<q$ to*

$$\dim(\mathrm{RM}(q,m,r))=\binom{m+r}{r}$$

Dowód to proste zliczanie, pozostawimy jako ćwiczenie.

Odległość

Lemat 10.4 (Schwartz–Zippel)**.** *Niech $f\in\mathbb{F}_q[X_1,\ldots,X_m]$ będzie niezerowym wielomianem stopnia $\deg(f)\le r$. Wtedy frakcja 0 wśród wszystkich wartościowań to najwyżej $\frac{r}{q}$, tj.:*

$$\frac{\left|\{\vec{v}\in\mathbb{F}_q^m\ :\ f(\vec{v})=0\}\right|}{q^m}\le\frac{r}{q}$$

Czyli odległość to przynajmniej $q^m-rq^{m-1}=(q-r)q^{m-1}$.

Twierdzenie 10.5. *Dla $r<q$ kod $\mathrm{RM}(q,m,r)$ ma wymiar $\dim(\mathrm{RM}(q,m,r))=\binom{m+r}{r}$ i odległość przynajmniej $\mathrm{d_H}(\mathrm{RM}(q,m,r))\ge(q-r)q^{m-1}$.*

Uwaga. Dla $m=1$ to jest stwierdzenie, że wielomian stopnia r ma $\le r$ pierwiastków.

Dla każdego m istnieją wielomiany, że to jest ścisłe.

Ale są wielomiany, dla których to nie jest ścisłe.

Uwaga. Tak naprawdę to ten Lemat zwykle mówi, że jeśli wybieramy wszystkie wartości z podzbioru S (losowo), to prawdopodobieństwo wynosi najwyżej $\frac{\deg(f)}{|S|}$. Nasz dowód to de facto pokazuje, ale nie będziemy się nad tym rozwodzić. W szczególności ciało nie musi być skończone, wystarczy, że bierzemy ze skończonego podzbioru.

Lematu 10.4. To jest równoważne temu, że jeśli wylosujemy punkt ewaluacji $\vec{v}$, to prawdopodobieństwo tego, że ustalone f zeruje się na nim, tj. $f(\vec{v})=0$, wynosi najwyżej $\frac{\deg(f)}{q}$.

Indukcja po $m\ge 1$. Dla $m=1$ to wynika z tego, ile pierwiastków może mieć wielomian jednej zmiennej.

Rozważmy $m>1$, zakładamy, że Lemat jest prawdziwy dla $m-1$. Zapisujemy f jako wielomian jednej zmiennej nad X_m ze współczynnikami, które same są wielomianami z $\mathbb{F}_q[X_1,\ldots,X_{m-1}]$.

$$f=\sum_{i=0}^{t}f_iX_m^i\ ,$$

gdzie każde $f_i(X_1,\ldots,X_{m-1})$ jest wielomianem z $\mathbb{F}_q[X_1,\ldots,X_{m-1}]$ i $\deg(f_i)\le r-i$ oraz t jest maksymalne takie, że $f_t\neq 0$. Nasz wybór $\vec{v}=(a_1,\ldots,a_m)$ przeprowadzamy w dwóch krokach: najpierw wybieramy $(a_1,\ldots,a_{m-1})$ a potem a_m.

Niech

$$f^{(a_1,\ldots,a_{m-1})}(X_m)=\sum_{i=0}^{t}f_i(a_1,\ldots,a_{m-1})X_m^i$$

Rozważamy dwa możliwe zdarzenia:

$$Y_1=\{a_1,\ldots,a_m\ :\ f_t(a_1,\ldots,a_{m-1})=0\}$$
$$Y_2=\{a_1,\ldots,a_m\ :\ f_t(a_1,\ldots,a_{m-1})\neq 0\text{ i }f^{(a_1,\ldots,a_{m-1})}(a_m)=0\}.$$

Zauważmy, że $Y_1 \cup Y_2$ pokrywa zbiór $\{(a_1, \ldots, a_m) : f(a_1, \ldots, a_m) = 0\}$. W szczególności, szukane przez nas prawdopodobieństwo to najwyżej $\mathbb{P}[Y_1] + \mathbb{P}[Y_2]$.

Z założenia indukcyjnego mamy

$$\mathbb{P}[Y_1] \leq \frac{r-t}{q}$$

bo $\deg(f_t) \leq r - t$ i $f_t \neq 0$.

Z drugiej strony, $f^{(a_1,\ldots,a_{m-1})}$ jest wielomianem jednej zmiennej i stopnia t, czyli $\mathbb{P}[Y_2] \leq \frac{t}{q}$. Co daje tezę. □

10.1.1 Porównanie z innymi kodami

Powyższe oszacowania, choć dość precyzyjne, nie są łatwe w analizowaniu i nie jest je prosto porównać ze znanymi nam kodami. Rozpatrzmy parę przypadków szczególnych:

Dla $m = 1$, $r = k - 1$ dostajemy kody Reeda-Solomona ewaluowane nad całym $\mathbb{F}_q$.

Dla $m, r = 1$ i $q = 2$ dostajemy *rozszerzone kody Hadamarda*: $\{v, \overline{v} : v \in C_{\text{Had}}\}$. (Ćwiczenie.)

10.2 Kody binarne: $q = 2$

Przejdźmy teraz do kodów binarnych (to jest oryginalna definicja kodów RM).

Lemat 10.6. *Niech $f \in \mathbb{F}_2[X_1, \ldots, X_m]$ będzie niezerowym wielomianem stopnia $m = \deg(f)$ spełniającym $\deg_{X_i}(f) \leq 1$. Wtedy*

$$|\{(a_1, \ldots, a_m) \in \mathbb{F}_2^m : f(a_1, \ldots, a_m) \neq 0\}| \geq 2^{m-\deg(f)} .$$

Dowód pozostawiamy jako ćwiczenie.

Czyli waga każdego niezerowego słowa kodowego to przynajmniej $2^{m-\deg(f)}$.

To ograniczenie jest ścisłe, dowód również pozostawiamy jako ćwiczenie.

Wymiar kodu jest stosunkowo łatwo policzyć.

Lemat 10.7. *Dla każdego $r \leq m$ wymiar kodu Reed-Mullera $\mathrm{RM}(2, m, r)$ wynosi $\sum_{i=0}^{r} \binom{m}{i}$.*

Wymiar to liczba jednomianów spełniających warunki na stopnie: ogólnie słowo kodowe odpowiada wybraniu podzbioru takich jednomianów. Zauważmy, że stopień względny wynosi 0 lub 1 dla każdej zmiennej, dlatego jednomian można identyfikować z podzbiorem zmiennych. Czyli liczymy podzbiory $\{X_1, \ldots, X_m\}$ o wielkości co najwyżej r.

Twierdzenie 10.8. *Dla każdego $r \leq m$ kody Reeda-Mullera $\mathrm{RM}(2, r, m)$ są kodami o długości kodu 2^m, wymiaru $\sum_{i=0}^{r} \binom{m}{i}$ i odległości 2^{m-r}.*

10.3 Przypadek ogólny

Policzenie wymiaru jest możliwe

Lemat 10.9. *Niech $0 \neq f \in \mathbb{F}_q[X_1, \ldots, X_m]$ będzie wielomianem niezerowym wielomianem o współczynnikach z $\mathbb{F}_q$ oraz zmiennych $X_1, \ldots, X_m$ spełniającym warunki: $\deg_{X_i}(f) \leq q - 1$ dla każdego $i \in \{1, \ldots, m\}$ oraz $\deg(f) \leq r$. Niech s, t będą (jedynymi) nieujemnymi liczbami całkowitymi takimi że*

$$t \leq q - 2 \text{ oraz } s(q-1) + t = r .$$

(tj. $t = r \bmod (q-1)$ i $s = (r-t)/(q-1)$). Wtedy

$$\left|\{\vec{v} \in \mathbb{F}_q^m : f(\vec{v}) \neq 0\}\right| \geq (q-t) \cdot q^{m-s-1} \geq q^{m-\frac{r}{q-1}} .$$

Wniosek 10.10. $\mathrm{d_H}(\mathrm{RM}(q,m,r)) \geq q^{m-\frac{r}{q-1}}$

Dowód jest podobny do poprzednich oszacowań, ale ma dużo więcej szczegółów technicznych i przypadków, nie pokażemy.

Oszacowanie wymiaru jest dużo trudniejsze.

Definicja 10.11. Oznaczmy $K_{q,m,r}$ — wymiar kodu $\mathrm{RM}(q,m,r)$.

Zdefiniujmy

$$K^+_{q,m,r} = \min\left(q^m, \binom{m+r}{r}\right)$$

$$K^-_{q,m,r} = \begin{cases} \max\left(\frac{q^m}{2}, q^m - K^+_{q,m,(q-1)m-r}\right) & \text{dla } r \geq \frac{(q-1)m}{2} \\ \max\left(\binom{m}{r}, \frac{1}{2}\left\lfloor\frac{2r+m}{m}\right\rfloor^m\right) & \text{dla } r < \frac{(q-1)m}{2} \end{cases}$$

Twierdzenie 10.12. *Dla pewnych stałych c_1, c_2 (gdzie $c_1 < 3,1$, $c_2 < 8,2$) zachodzi*

$$K^-_{q,m,r} \leq K_{q,m,r} \leq K^+_{q,m,r} \leq c_1\left(K^-_{q,m,r}\right)^{c_2} \ .$$

Ograniczenie górne jest trywialne. Dolne yo kilka osobnych konstrukcji i sporo szacowań.

10.4 Dekodowanie dla $q = 2$ (algorytm Reeda, dekodowanie większością)

Idea algorytmu opiera się na następującym fakcie

Lemat 10.13. *Niech $P \in \mathbb{F}_2[X_1, \ldots, X_m]$ będzie wielomianem o współczynnikach z $\mathbb{F}_2$ stopnia $\deg(P) = r$, niech również $C \in \mathbb{F}_2$ będzie współczynnikiem dla jednomianu $\prod_{i=1}^r X_i$ w P. Wtedy dla każdego $\vec{b} \in \mathbb{F}_2^{m-r}$ zachodzi*

$$\sum_{\vec{a}\in\mathbb{F}_2^r} P(\vec{a},\vec{b}) = C \ .$$

Zanim podamy dowód, zastanówmy się nad konsekwencjami. Na pierwszy rzut oka wydawałoby się, że ten Lemat przydaje się tylko w przypadku, gdy nie ma błędów — no bo skąd ma być wiadomo, że wartości dla konkretnych $\vec{a}, \vec{b}$ są dobre? Ważne jest to, że wynik C powinien być taki sam dla *każdego możliwego* $\vec{b}$. Takich wektorów jest $2^{m-r} = \mathrm{d_H}(\mathrm{RM}(2,m,r))$. Jeśli błędów jest mniej niż połowa odległości, to dla *większości* możliwych $\vec{b}$ ich sumy nie będą zawierały żadnego błędu. Czyli poprawny wynik wychodzi dla większości podstawień $\vec{b}$

Widać, że wybór jednomianu $\prod_{i=1}^r X_i$ jest arbitralny i argument jest prawdziwy dla każdego jednomianu stopnia r. A potem będziemy przez indukcje schodzić w dół z wartością r.

Dowód. Zdefiniujmy $P_{\vec{b}}(X_1, \ldots, X_r) = P(X_1, \ldots, X_r, \vec{b})$. Zauważmy, że współczynnik przy jednomianie $\prod_{i=1}^r X_i$ w P i $P_{\vec{b}}$ jest taki sam i jest to jedyny jednomian stopnia r. (Tu jest ważne, że $\deg(P) = r$, bo inaczej coś mogłoby się uprościć do $\prod_{i=1}^r X_i$. Czyli

$$P_{\vec{b}} = C\prod_{i=1}^r X_i + g \ ,$$

gdzie $\deg(g) < r$. Chcemy pokazać, że C to

$$\sum_{\vec{a}\in\mathbb{F}_2^r} P_{\vec{b}}(\vec{a}) = \sum_{\vec{a}\in\mathbb{F}_2^r} C\prod_{i=1}^r X_i(\vec{a}) + g(\vec{a})$$

Najpierw zauważamy, że pierwsza składnik jest równy C, bo $\prod_{i=1}^{r} X_i(\vec{a})$ jest niezerowy tylko dla $\vec{a}$ będącego ciągiem jedynek a dla samych jedynek ewaluuje sie do 1 i tym samym całość do C.

Pozostaje pokazać, że jeśli g jest wielomianem r zmiennych oraz $\deg(r) < r$ to

$$\sum_{\vec{a} \in \mathbb{F}_2^r} g(\vec{a}) = 0$$

Pozostawiamy to jako ćwiczenie, de facto użyliśmy tego już wcześniej, przy liczeniu odległości kodu RM. □

Oznaczenia: Dla prostoty, jeśli $S \subseteq \{1, \ldots, m\}$ to przez X_S oznaczamy jednomian $\prod_{i \in S} X_i$. Dla $S \subseteq \{1, \ldots, m\}$, gdzie $|S| = t$ i wektorów $\vec{a} \in \mathbb{F}_2^t$ i $\vec{b} \in \mathbb{F}_2^{m-t}$ przez $(S \leftarrow \vec{a}, \overline{S} \leftarrow \vec{b})$ oznaczamy wektor w którym współrzędne z S są dane przez $\vec{a}$ pozostałe przez $\vec{b}$. Dla $S \subseteq \{1, \ldots, m\}$, gdzie $|S| = t$ i wektorów $a \in \mathbb{F}_2^t$ przez $f(S \leftarrow \vec{a}, \overline{S} \leftarrow \vec{b})$ oznacza obliczenie f na odpowiednim wektorze.

Algorytm
Dane: $r < m$, $0 \leq e \leq \frac{2^{m-r}}{2}$, funkcja $f : \mathbb{F}_2^m \to \mathbb{F}_2$.
Szukane: wielomian P taki że $\deg(P) \leq r$, $|\{\vec{x} \in \mathbb{F}_2^m \ : \ f(\vec{x}) \neq P(\vec{x})\}| \leq e$. Algorytm iteruje po t od r do 0 w dół. Dla ustalonej wartości t przegląda wszystkie możliwe $S \subseteq \{1, \ldots, m\}$ mocy t i ustala wartość współczynnika przy X_S na podstawie Lematu 10.13.

Algorytm 2 Algorytm Reeda

```
1: P ← 0
2: for t ← r . . 1 do
3:     for S ⊆ {1, . . . , m}: |S| = t do
4:         for b⃗ ∈ F_2^{m−t} do
5:             C_{S,b⃗} ← Σ_{a⃗ ∈ F_2^t} (f − P)(S ← a⃗, S̄ ← b⃗)
6:         C_S ← majority_{b⃗} {C_{S,b⃗} : b⃗ ∈ F_2^t}
7:         P ← P + C_S X_S
8: return P
```

10.4.1 Analiza

Lemat 10.14. *Jeśli dane na wejściu $f : \mathbb{F}_2^m \to \mathbb{F}_2$, które nie zgadza się z pewnym wielomianem $Q \in \mathbb{F}_2[X_1, \ldots, X_m]$ stopnia najwyżej r na co najwyżej $e < \frac{1}{2} \cdot 2^{m-r}$ punktach, to algorytm Reeda zwróci Q.*

Dowód. Dowód opiera się na następujące obserwacji:

Fakt 10.15. *Jeśli wielomian f oraz $Q \in \mathrm{RM}(2, m, t)$ różnią się na $< \frac{1}{2} 2^{m-r}$ pozycjach, dla $r \geq t$, to algorytm poprawnie oblicza współczynnik przy X_S dla $|S| = t$.*

Dowód wynika z Lematu 10.13: mówimy, że $\vec{b}$ jest *błędne*, jeśli dla pewnego $\vec{a}$ mamy, że $f(S \leftarrow \vec{a}, \overline{S} \leftarrow \vec{b}) \neq Q(S \leftarrow \vec{a}, \overline{S} \leftarrow \vec{b})$. Jest mniej niż $\frac{1}{2} \cdot 2^{m-r}$ błędnych b, bo kazdy błąd liczy się dla jednego $\vec{b}$, czyli w takim razie dla większości z nich wyliczymy taką wartość, jak dla Q, czyli z Lematu 10.13 jest to współczynnik przy X_S (zauważmy, że $\deg(Q) = t \leq r$, ale to jest OK.)

Zauważmy też, że przy tych założeniach $f - C_S X_S$ i $Q - C_S X_S$ dalej różnią się na tej samej liczbie argumentów, co f i Q.

W takim razie dowód przebiega przez indukcję: zakładamy, że na początku mamy wielomian Q różniący się od wejścia na $< 2^{m-r}/2$ argumentach. W każdej chwili mamy, że $f - P$ oraz $Q - P$ różnią się na najwyżej tylu argumentach oraz P i Q zgadzaja sie jednomianach, które rozważył już algorytm. Na koniec $P = Q$ i dostajemy poprawny wynik. □

Czas działania algorytmu jest łatwy do oszacowania przez $\mathcal{O}(n^2)$, pozostawiamy to jako ćwiczenie. Da się też trochę lepiej. Co daje następujące twierdzenie.

Twierdzenie 10.16. *Dla każdego $0 \leq r < m$, oraz $d = 2^{m-r}$ dekoder Reeda koryguje do $\frac{d}{2} - 1$ błędów w kodzie Reed-Mullera $RM(2, m, r)$. Robi to w czasie $\mathcal{O}(n^2)$, gdzie $n = 2^m$ jest długością kodu, a jest jego odległością.*

10.5 Dekodowanie przez redukcję do kodów RS.

Pokażemy teraz algorytm, używa bardziej wyrafinowanego pomysłu algebraicznego, ale wynikająca z niego redukcja jest w zasadzie trywialna, w tym sensie, że prawie nic nie robi.

Istotą redukcji jest naturalna bijekcja pomiędzy przestrzenią wektorową $\mathbb{F}_q^m$ i ciałem $\mathbb{F}_{q^m}$. Ta bijekcja naturalnie rozszerza się do bijekcji z przestrzeni funkcji $\{f : \mathbb{F}_q^m \to \mathbb{F}_q\}$ na przestrzeń funkcji $\{f : \mathbb{F}_{q^m} \to \mathbb{F}_q\} \subseteq \{f : \mathbb{F}_{q^m} \to \mathbb{F}_{q^m}\}$. Z punktu widzenia algorytmu ważne jest, że bijekcja dziala na dziedzinie a nie na obrazach. W ten sposób kody Reed-Mullera zostają przekształcone w podkod kodu Reeda-Solomona i błedne słowa kodowe Reeda-Mullera są mapowane na błędne słowa kodowe Reeda-Solomona. Następnie analizujemy, jaka odległość tak uzyskanego kodu Reed-Solomona, lub równoważnie, górna granica stopnia wielomianów G uzyskanych w wyniku zastosowania bijekcji do $g \in \mathrm{RM}(q, m, r)$. Okazuje się, że bijekcja zachowuje odległość i tym samym tak algorytmy korygujące kod Reed-Solomona. do połowy jego odległości robi to samo dla kodu Reed-Mullera.

10.5.1 Algorytm z lotu ptaka

Algorytm bedzie korzystał z ustalonej bijekcji ($\mathbb{F}_q$-liniowej, potem powiemy, co to znaczy)

$$\Phi : \mathbb{F}_{q^m} \to \mathbb{F}_q^m$$

Wejście, czylik kod Reeda Mullera, potraktujemy jako daną funkcję $f : \mathbb{F}_q^m \to \mathbb{F}_q$

Stworzymy słowo dla algorytmu RS dla $n = q^m$ (czyli ewaluujemy we wszystkich elementach ciała $\mathbb{F}_{q^m}$).

Stworzymy funkcje $F : \mathbb{F}_{q^m} \to \mathbb{F}_q \subseteq \mathbb{F}_{q^m}$.

Dla $u \in \mathbb{F}_{q^m}$ ustalamy $F(u) \leftarrow f(\Phi(u))$.

Następnie ten wielomian dekodujemy (do P) i przypisuje z powrotem, tj. dla $u \in \mathbb{F}_{q^m}$ $p(\Phi^{-1}(u) \leftarrow P(u)$

Uwaga. Trzeba sprawdzić, że to ten kod RS jest dobrze zdefiniowany, algorytm faktycznie daje element w ciele $\mathbb{F}_q$, czy to dobrze działa, ile błędów koryguje, itp.

Uwaga. Zwróćmy uwagę, że dekoder zwraca wielomian $p \in \mathbb{F}_q[X_1, \ldots, X_m]$ jako ciąg wartości we wszystkich punktach. Chcemy reprezentację w postaci ciągu współczynników, co można uzyskać poprzez interpolację. W jaki sposób dokładnie ją przeprowadzić — ćwiczenie.

10.5.2 Bijekcja z $\mathbb{F}_{q^m} \to \mathbb{F}_q^m$

Definicja 10.17 ($\mathbb{F}$-liniowa bijekcja)**.** Funkcję $\Phi : \mathbb{F}_{q^m} \to \mathbb{F}_q^m$ nazywamy bijekcją $\mathbb{F}_q$-liniową, jeżeli

- Φ jest bijekcją.
- Φ jest liniowe nad $\mathbb{F}_q$, tj. dla każdych $\alpha, \beta \in \mathbb{F}_q$ i $u, v \in \mathbb{F}_{q^m}$ zachodzi $\Phi(\alpha u + \beta v) = \alpha\Phi(u) + \beta\Phi(v)$.

Zauważmy, co będzie też używane potem, że $\mathbb{F}_q \subseteq \mathbb{F}_{q^m}$ i dlatego wyrażenia takie αu dla $\alpha \in \mathbb{F}_q$ i $u \in \mathbb{F}_{q^m}$ są dobrze zdefiniowane.

Funkcję $\Phi : \mathbb{F}_{q^m} \to \mathbb{F}_q^m$ będziemy też traktować jako wektor funkcji $(\Phi_1, \ldots, \Phi_m)$, gdzie $\Phi_i : \mathbb{F}_{q^m} \to \mathbb{F}_q$. Ponadto, każde Φ_i jest funkcją $\mathbb{F}_q$-liniową. Co więcej, skoro $\Phi_i : \mathbb{F}_{q^m} \to \mathbb{F}_q \subseteq \mathbb{F}_{q^m}$, to możemy ją traktować jako funkcję $\mathbb{F}_{q^m} \mapsto \mathbb{F}_{q^m}$ i tym samym jako wielomian (pokazaliśmy dla ciała $\mathbb{F}_2$, ale dowód jest ten sam); wielomian ten ma jednak dodatkową własność: każda wartość jest z $\mathbb{F}_q$. Scharakteryzujmy najpierw takie funkcje (co jest klasycznym w matematyce wynikiem).

Definicja 10.18. Funkcja ślad $\mathrm{tr}_{\mathbb{F}_{q^m} \to \mathbb{F}_q}$ jest zadana przez ewaluację wielomianu $\mathrm{tr}(X) = X + X^q + X^{q^2} + X^{q^3} + \cdots + X^{q^{m-1}}$ w punkcie, tj.:

$$\mathrm{tr}(\alpha) = \sum_{i=0}^{m-1} \alpha^{q^i} \ .$$

Lemat 10.19. *Ślad jest $\mathbb{F}_q$-liniowa, tzn. dla każdych $\alpha \in \mathbb{F}_q$ $u, v \in \mathbb{F}_{q^m}$ zachodzi*

$$\mathrm{tr}(\alpha v) = \alpha v \quad \mathrm{tr}(u + v) = \mathrm{tr}(u) + \mathrm{tr}(v)$$

oraz obraz zawarty jest w $\mathbb{F}_q$.

Dowód. $\mathbb{F}_q$-liniowość wynika z tego, że

$$(u + v)^{q^i} = u^{q^i} + v^{q^i}$$

oraz

$$\alpha^{q^i} = \alpha$$

pierwsze wynika z tego, że ciało jest charakterystyki q, zaś drugie z tego, że α jest elementem ciała $\mathbb{F}_q$ i tym samym $\alpha^q = \alpha$.

Co do wartości: jako ćwiczenie zostawmy fakt, że elementy ciała $\mathbb{F}_q$ w $\mathbb{F}_{q^m}$ to dokładnie te spełniające równanie $X^q = X$. Również jako ćwiczenie pozostawimy sprawdzenie, że faktycznie wartości śladu spełniają to równanie. □

Pokazaliśmy już, że ślad jest funkcją $\mathbb{F}_q$-liniową, okazuje się, że przy jego pomocy można zdefiniować wszystkie takie funkcje, potrzebny jest tylko dodatkowy parametr z $\mathbb{F}_{q^m}$.

Lemat 10.20. *Funkcja $L : \mathbb{F}_{q^m} \to \mathbb{F}_q$ jest $\mathbb{F}_q$-liniowa wtedy i tylko wtedy, gdy istnieje $\lambda \in \mathbb{F}_{q^m}$, taka że*

$$L(\beta) = \mathrm{tr}(\lambda\beta) \ .$$

Dowód. Po pierwsze należy zauważyć, że

$$f_\lambda(\beta) = \mathrm{tr}(\lambda\beta)$$

jest oczywiście $\mathbb{F}_q$-liniowa, co wynika z $\mathbb{F}_q$-liniowości śladu.

Aby pokazać, że wszystkie funkcje $\mathbb{F}_q$-liniowe są tej postaci, zliczymy je wszystkie.

Pokażmy najpierw, że dla różnych $\lambda, \lambda' \in \mathbb{F}_{q^m}$ funkcje f_λ i $f_{\lambda'}$ są różne. Popatrzmy na funkcję $f_\lambda - f_{\lambda'}$. Jest ona wielomianem stopnia najwyżej q^{m-1} i wielomian ten jest niezerowy,

bo przy X ma współczynnik $\lambda - \lambda'$. W takim razie ma najwyżej q^{m-1} pierwiastków, czyli na jakimś z q^m elementów ciała się nie zeruje.

Teraz pozostało pokazać, że takich funkcji jest najwyżej q^m. Ale to wiadomo z podstawowych faktów z algebry liniowej: $\mathbb{F}_{q^m}$ można traktować jako przestrzeń liniową wymiaru m nad $\mathbb{F}_q$. Wtedy $L : \mathbb{F}_{q^m} \to \mathbb{F}_q$ będące $\mathbb{F}_q$ liniowe indukuje przekształcenie liniowe $L : \mathbb{F}_q^m \to \mathbb{F}_q$. To przekształcenie to funkcjonał liniowy i wiemy, że jest ich q^m (albo można po prostu zliczyć macierze). □

Rozszerzymy teraz charakteryzację do bijekcji liniowych.

Lemat 10.21. *Funkcja $\Phi : \mathbb{F}_{q^m} \to \mathbb{F}_q^m$ jest bijekcją $\mathbb{F}_q$-liniową wtedy i tylko wtedy, gdy $\Phi = (\Phi_1, \ldots, \Phi_m)$ dla $\Phi_i : \mathbb{F}_{q^m} \to \mathbb{F}_q$ i każde Φ_i jest postaci $\Phi_i(\alpha) = \operatorname{tr}(\lambda_i \alpha)$, dla $\lambda_1, \ldots, \lambda_m \in \mathbb{F}_{q^m}$, które są liniowo niezależne nad $\mathbb{F}_q$.*

W szczególności dla Φ_i jak wyżej zachodzi $\deg(\Phi_i) = q^{m-1}$.

Dowód. $\Rightarrow$

Postać każdej z Φ_i oraz stopień wynika z Lematu 10.20. To, że dla liniowo zależnych $\lambda_1, \ldots, \lambda_m$ nie wychodzi bijekcja pozostawiamy jako ćwiczenie.

$\Leftarrow$ Twierdzimy, że dla liniowo niezależnych (nad $\mathbb{F}_{q^m}$) liczb $\lambda_1, \ldots, \lambda_m$ tak zdefiniowana funkcja Φ jest bijekcją liniową. Liniowość wynika z liniowość śladu, pozostaje zweryfikować, że jest to bijekcja. Ponieważ dziedzina i przeciwdzidzina są tej samej mocy, wystarczy pokazać, że Φ jest „na". Rozważmy zbiór

$$S = \operatorname{Im} \Phi = \{\Phi(u) \ : \ u \in \mathbb{F}_{q^m}\} \subseteq \mathbb{F}_q^m$$

Ponieważ Φ jest $\mathbb{F}_q$-liniowa, $S \leq \mathbb{F}_q^m$ (jako podprzestrzeń liniowa nad ciałem $\mathbb{F}_q$). Jeśli $S \neq \mathbb{F}_q^m$, (tj. jeśli Φ nie jest „na"), to elementy S spełniają pewną nietrywialną równość liniową, tzn. istnieje $(\alpha_1, \ldots, \alpha_m) \in \mathbb{F}_q^m \setminus \vec{0}$ (bierzemy z $S^\perp$) takie że

$$\sum_{i=0}^{m} \alpha_i \beta_i = 0 \text{ dla każdych } \beta_1, \ldots, \beta_m \in S \ .$$

Rozważmy

$$\begin{aligned} \sum_{i=1}^{m} \alpha_i \Phi_i(Z) &= \sum_{i=1}^{m} \alpha_i \operatorname{tr}(\lambda_i Z) \\ &= \operatorname{tr}\left(\underbrace{\left(\sum_{i=1}^{m} \alpha_i \lambda_i \right)}_{\neq 0} Z \right) \end{aligned}$$

Z jednej strony, jest to niezerowy wielomian stopnia co najwyżej q^{m-1} (bo mnożnik przed Z to niezerowa kombinacja λ_i, czyli z założenia o niezależności — liczba niezerowa), a z drugiej strony ewaluuje się do 0 na każdym $u \in \mathbb{F}_{q^m}$, sprzeczność.

Czyli Φ jest na, a więc jest bijekcją $\mathbb{F}_q$-liniową. □

Naszym celem jest zastosowanie bijekcji liniowej $\Phi : \mathbb{F}_{q^m} \to \mathbb{F}_q^m$ (każda nadaje się równie dobrze) by skonwertować funkcje, których dziedziną jest $\mathbb{F}_q^m$ (co ma miejsce w przypadku kodów Reed-Mullera) do funkcji, których dziedziną jest $\mathbb{F}_{q^m}$, co ma miejsce dla kodów Reeda-Solomona. W szczególności, jeśli $f : \mathbb{F}_q^m \to \mathbb{F}_q$ i $\Phi : \mathbb{F}_{q^m} \to \mathbb{F}_q^m$ to oznaczamy $f \circ \Phi : \mathbb{F}_{q^m} \to \mathbb{F}_q$ na ich złożenie zdefiniowana naturalnie jako $(f \circ \Phi)(u) = f(\Phi(u))$. Kluczem użyteczności tego przekształcenia

jest analiza, w jaki sposób zwiększa (lub nie) ono stopień wielomianów. Przypomnijmy, że dla funkcji $F : \mathbb{F}_{q^m} \to \mathbb{F}_{q^m}$, jej stopień $\deg(F)$ jest zdefiniowany jako stopień (jedynego) wielomianu $P \in \mathbb{F}_{q^m}[Z]$ spełniającego $\deg(P) < q^m$, takiego że $P(u) = F(u)$ dla wszystkich $u \in \mathbb{F}_{q^m}$. Od tej pory naszym głównym wyzwaniem jest zrozumienie problemu:
Jak duży może być stopień $f \circ \Phi$ w stosunku do stopnia $f : \mathbb{F}_q^m \to \mathbb{F}_q$?

Zdefiniujemy najpierw odpowiadający temu parametr i zanalizujemy go.

Definicja 10.22. Dla q — liczby pierwszej i nieujemnych liczb całkowitych m, r, niech stopień rozszerzenie r nad $\mathbb{F}_{q^m}$, oznaczany przez $R_{q,m}(r)$, jest maksymalnym stopniem $p \circ \Phi$ po wszystkich możliwych wyborach wielomianu m zmiennych $p \in \mathbb{F}_q[X_1, \ldots, X_m]$ stopnia najwyżej r (lub równoważnie: wszystkich odpowiadający funkcjach $p : \mathbb{F}_q^m \to \mathbb{F}_q$) oraz po wszystkich możliwych bijekcjach $\mathbb{F}_q$-liniowych $\Phi : \mathbb{F}_{q^m} \to \mathbb{F}_q^m$.

Algorytm i analiza są teraz naturalne z dokładnością do ustalenia wartości $R_{q,m}(r)$.

Pokażemy najpierw, że ten algorytm poprawia $e < \frac{q^m - R_{q,m}(r)}{2}$ błędów.

Lemat 10.23. *$f : \mathbb{F}_q^m \to \mathbb{F}_q$ będzie dowolną funkcją i niech $g : \mathbb{F}_q^m \to \mathbb{F}_q$ będzie wielomianem stopnia $\deg(g) \leq r$ takim że $\left|\{u \in \mathbb{F}_q^m \ : \ f(u) \neq g(u)\}\right| \leq e < (q^m - R_{q,m}(r))/2$. Wtedy algorytm oparty na redukcji RM do RS zwraca g.*

Dowód. Niech $G = g \circ \Phi$ oraz $F = f \circ \Phi$. W takim razie $\deg(G) \leq R_{q,m}(r)$ i

$$|\{v \in \mathbb{F}_{q^m} \ : \ F(v) \neq G(v)\}| \leq e < (q^m - R_{q,m}(r))/2 \ .$$

Łatwo zauważyć, że kod dla g zostanie przekształcony na kod dla G przy czym $G \in \mathrm{RS}(\mathbb{F}_{q^m}, q^m, R_{m,q}(r) + 1)$, tj. $N = q^m$ oraz $K = R_{m,q}(r) + 1$. Odległość takiego kodu Reeda-Solomona to $N - K + 1 = q^m - R_{q,m}(r)$ dostajemy, że e jest mniejsza niż połowa odległości kodu. Czyli G jest jedynym wielomianem w odległości mniejszej niż polowa odległości kodu od F, więc dekoder dla kodów RS zwraca $P = G$. Czyli $p = P \circ \Phi^{-1} = G \circ \Phi^{-1} = g$. □

10.5.3 Analiza extension degree

Pokażmy najpierw dwa proste warianty jako wprawkę.

Lemat 10.24.

$$R_{q,m}(r) \leq r \cdot q^{m-1} \ .$$

Dowód. Dowód wynika bezpośrednio z definicji i faktu, że funkcje liniowe są wielomianami stopnia q^{m-1}.

Niech $p \in \mathbb{F}_q[X_1, \ldots, X_m]$ będzie miał stopień $\leq r$ i niech $\Phi = (\Phi_1, \ldots, \Phi_m)$ będzie $\mathbb{F}_q$-liniową bijekcją. Każdy z Φ_i jest wielomianem stopnia q^{m-1}, mamy że $p \circ \Phi = p(\Phi_1(Z), \ldots, \Phi_m(Z))$ jest wielomianem stopnia co najwyżej $r \cdot q^{m-1}$. □

Wniosek 10.25. Jeśli $r < q$ to redukcja do kodów RS dekoduje kody $\mathrm{RM}(q, m, r)$ dla $< \frac{(q-r)q^{m-1}}{2}$ błędów; czyli do połowy najlepszego ograniczenia na odległość kodów RM.

Dowód. Z Lematu 10.10 mamy, że odległość kodu Reed-Mullera $RM(q, m, r)$ wynosi przynajmniej $(q - t) \cdot q^{m-1}$.

Oszacujmy:

$$\begin{aligned} \frac{q^m - R_{q,m}(r)}{2} &\geq \frac{q^m - r \cdot q^{m-1}}{2} \\ &= \frac{(q-r) \cdot q^{m-1}}{2} \end{aligned}$$

W takim razie dla $e < \frac{(q-r) \cdot q^{m-1}}{2} \leq \frac{q^m - R_{q,m}(r)}{2}$ z Lematu 10.23 dostajemy, że algorytm dekoduje poprawnie. □

Pozostało więc rozpatrzyć przypadek dla $r > q$. Tu będziemy musieli nietrywialnie skorzystać z tego, że liczymy stopień dla funkcji. Tj. wielomian, który nam wyjdzie, może mieć duży stopień, ale można go zredukować używając równości (w ciele $\mathbb{F}_{q^m}$) $Z^{q^m} = Z$, która jest prawdziwa dla każdego podstawienia pod Z. Formalnie, będziemy brali wielomiany modulo $Z^{q^m} - Z$, ich stopień będzie wtedy mniejszy niż q^m.

Lemat 10.26. *Niech $r = s(q-1)+t$ gdzie $0 \leq t < q-1$. Wtedy $R_{q,m}(r) = q^m - (q-t)q^{m-s-1}$.*

Twierdzenie 10.27. *Dla każdej liczby pierwszej q oraz liczb całkowitych $m \geq 1$, $0 \leq r < m(q-1)$, dekoder redukujący do kodów RS poprawia błędy w $\mathrm{RM}(q,m,r)$ do połowy minimalnej odległości (czyli do $\leq ((q-r)q^{m-1}-1)/2$).*

Dowód. Niech $r = s(q-1)+t$, gdzie $0 \leq t < q-1$. Wiemy, że minimalna odległość to $(q-t)\cdot q^{m-s-1}$. Z Lematu 10.23 mamy, że algorytm dekoduje jeśli $e < (q^m - R_{m,q}(r))/2$ Wstawiając oszacowanie z Lematu 10.26 dostajemy $= (q-t)\cdot q^{m-s-1}/2$, tj. do połowy minimalnej odległości $\mathrm{RM}(q,r,m)$. □

Pozostaje pokazać Lemat 10.26, Zanim przejdziemy do definicji oraz dowodu, zastanówmy się przez chwilę nad operacją, którą wykonujemy, czyli braniem wielomianów modulo $Z^{q^m} - Z$.

Zauważmy najpierw prostą równość:

$$\begin{aligned} Z^{q^m} &\equiv_{Z^{q^m}-Z} Z && \Longrightarrow \\ Z^{q^{m+k}} &\equiv_{Z^{q^m}-Z} Z^{q^k} && \Longrightarrow \\ Z^{a+q^{m+k}} &\equiv_{Z^{q^m}-Z} Z^{a+q^k} && (10.1) \end{aligned}$$

Zauważmy, że ostatnia linia oznacza, że jeśli popatrzymy na wykładnik w potędze Z i na jego zapis pozycyjny w podstawie q, to możemy z dowolnej pozycji zabrać 1 i przesunąć ją m pozycji niżej (tam się może dodać i być przeniesienie). Taki proces musi się zakończyć (na wykładniku mniejszym niż q^m) i wynik jest jednoznaczny, bo dwa wielomiany stopnia mniejszego niż q^m równe modulo $Z^{q^m} - Z$ sa równe.

W takim razie gdy patrzymy na równość modulo $Z^{q^m} - Z$ to zamiast patrzeć na stopień, lepiej jest patrzeć na sumę cyfr w zapisie o podstawie q tego stopnia. To uzasadnia poniższą definicję.

Definicja 10.28 (q-stopień)**.** Dla liczby całkowitej d, niech $d_0, \ldots, d_m, \ldots$ będzie jej reprezentacją w systemie q-pozycyjnym, tj. $d = \sum_{i\geq 0} d_i q^i$. Dla jednomianu Z^d jego *q-stopień* $\deg_q(Z^d)$ to $\sum_{i\geq 0} d_i$. Rozszerzamy na dowolny wielomian w naturalny sposób, tj. dla wielomianu $p(Z) = \sum_d c_d Z^d$ niech $\deg_q(p) = \max\{\deg_q(Z^d) \,:\, c_d \neq 0\}$.

Na przykład $\deg_2(X^8 + X^3 + X + 1) = \deg_2(X^3) = 2$.

Poniżej podajemy kilka podstawowych własności q-stopnia. Intuicyjnie poniższy lemat pokazuje, że tak zdefiniowany stopień zachowuje się tak jak zwykły stopień, jeśli chodzi o dodawanie, mnożenie i branie reszt modulo wielomian $Z^q - Z$. Zwracamy uwagę, że własności 1 i 2 są naturalne i proste, część 3 zachodzi tylko dla pewnych wielomianów; co jest naturalne, bo nasza definicja miała pasować do brania modulo $Z^{q^m} - Z$ i nie jest prawdziwa dla modulo dowolny wielomian. Część 4 pozwala powiązać q-stopień z prawdziwym stopniem; będzie to przydatne, gdy spróbujemy związać ograniczyć z góry stopień $f \circ \Phi \bmod Z^{q^m} - Z$.

Lemat 10.29. *Dla każdych $\alpha, \beta \in \mathbb{F}_{q^m}$ i $P, P_1, P_2 \in \mathbb{F}_{q^m}[Z]$ zachodzą następujące własności:*

1. $\deg_q(\alpha P_1 + \beta P_2) \leq \max\{\deg_q(P_1), \deg_q(P_2)\}$.
2. $\deg_q(P_1 \cdot P_2) \leq \deg_q(P_1) + \deg_q(P_2)$.

3. $\deg_q(P \bmod (Z^{q^m} - Z)) \leq \deg_q(P)$. *(Należy zwrócić uwagę, że w tym przypadku prawdziwy stopień* $\deg(P)$ *może być ściśle większy niż* q^m*.)*

4. *Jeśli* $\deg(P) < q^m$ *i* $\deg_q(P) = s(q-1) + t$ *dla* $0 \leq t < q-1$ *to*

$$\deg(P) \leq q^m - (q-t)q^{m-s-1}$$

Dowód. Część 1 wynika bezpośrednio z definicji, ponieważ każdy jednomian z $\alpha P_1 + \beta P_2$ znajdują się w sumie jednomianów P_1 i P_2 (ignorujemy współczynniki!).

W takim razie wystarczy, że pokażemy pozostałe własności dla jednomianów. Dla 2 dla jednomianów chcemy pokazać, że $\deg_q(Z^d \cdot Z^e) \leq \deg_q(Z^d) + \deg_q(Z^e)$. Niech $d = \sum_i d^i q^i$, $e = \sum_i e_i q^i$. Niech $f = d + e = \sum_i f_i q^i$. Łatwo sprawdzić, że

$$\sum_j f_j \leq \sum_j (d_j + e_j)$$

czyli również

$$\deg_q(Z^{d+e}) \leq \deg_q(Z^d) + \deg_q(Z^e) \ .$$

W części 3 ponownie skupiamy się na jednomianie Z^d, niech $d = \sum_i d_i q^i$. Przypomnijmy, że $Z^d \bmod (Z^{q^m} - Z)$ można obliczyć przy użyciu serii kroków jak w (10.1), tj. na zmodyfikowaniu wykładnika poprzez odjęcie 1 od $d_{m+k} > 0$ i dodanie 1 do d_k (to drugie może spowodować „przekręt licznika", jeśli $d_k = q-1$, co oczywiście może wywołać kaskadę takich przekręceń). Niech d' oznacza stopień po wykonaniu jednej takie operacji, w dowolnym miejscu. Wtedy

$$\deg_q(Z^d) \geq \deg_q(Z^{d'})$$

bo suma cyfr w zapisie o podstawie q nie rośnie (a może spaść o wielokrotność $q-1$). Czyli w ogólności jesteśmy w stanie przeprowadzić Z^d do $Z^d \bmod (Z^{q^m} - Z)$ nie zwiększając q-stopnia.

Przechodzimy do własności 4, w której porównujemy (prawdziwy) stopień wielomianu z jego q-stopniem. Przypominamy, że wystarczy pokazać dla jednomianu. Niech $q^m > \sum_i^{m-1} d_i q^i$. Liczby $d_0, \ldots, d_{m-1}$ spełniają warunek $\deg_q(d) = \sum_i d_i \leq s(q-1) + t$, gdzie $0 < t < q-1$. Wtedy d jest maksymalizowane dla $d_{m-1} = \cdots = d_{m-s} = (q-1)$ i $d_{m-s-1} = t$. Wtedy

$$d + (q-t)q^{m-s-1} = q^m$$

Wystarczy popatrzeć na zapis pozycyjny o podstawie q: dodajemy na ostatniej niezerowej pozycji do przeniesienia i potem jest przeniesienie na każdej pozycji. A to daje tezę. □

Następny Lemat porównuje stopień funkcji wielu zmiennych $f : \mathbb{F}_q^m \to \mathbb{F}_q$ z q-stopnia $f \circ \Phi$ dla dla q-liniowej bijekcji Φ.

Lemat 10.30. *Dla każdego wielomianu* $p \in \mathbb{F}_q[X_1, \ldots, X_m]$ *i każdej bijekcja* $\mathbb{F}_q$*-liniowej zachodzi*

$$\deg_q(p \circ \Phi) \leq \deg(p)$$

Dowód. Dzięki Lematowi 10.29(1) wystarczy udowodnić lemat dla jednomianu. Weźmy jednomian $M(X_1, \ldots, X_m) = \prod_{i=1}^m X_i^{r_i}$, gdzie $\sum_{i=1}^m r_i = r$. Ustalym również $\mathbb{F}_q$-liniową bijekcję $\Phi = (\Phi_1, \ldots, \Phi_m)$. Niech $\overline{M}$ będzie wielomianem jednej zmiennej Z zdefiniowanym jako $\overline{M} = M \circ \Phi \bmod (Z^q - Z)$. Wtedy

$$(M \circ \Phi)(Z) = \prod_{i=1}^m \Phi_i(Z)^{r_i} \ .$$

Zauważmy też, że dla każdego i zachodzi

$$\deg_q(\Phi i(Z)) = 1 \ .$$

Z Lematu 10.29(2) mamy

$$\deg_q(M \circ \Phi) = \deg_q \left(\prod_{i=1}^{m} \Phi_i(Z)^{r_i} \right) \leq \sum_{i=1}^{m} r_i = r \ .$$

I ostatecznie z Lematu 10.29(3)

$$\begin{aligned} \deg_q(\overline{M}) &= \deg_q \left(M \circ \Phi \mod (Z^{q^m} - Z) \right) \\ &\leq \deg_q (M \circ \Phi) \\ &\leq r \end{aligned}$$

□

Możemy teraz przystąpić do dowodu Lematu 10.26,

Dowód. Dla przypomnienia, chcemy pokazać, że

$$R_{q,m}(s(q-1)+t) = q^m - (q-t)q^{m-s-1} \ .$$

Pokażemy nierówność

$$R_{q,m}(s(q-1)+t) \leq q^m - (q-t)q^{m-s-1} \ .$$

Równość wynika wtedy z wyliczenia wartości $R_{q,m}$ dla konkretnych wielomianów i jest to ćwiczenie.

Ustalmy wielomian $p \in \mathbb{F}_q[X_1, \ldots, X_m]$ stopnia co najwyżej $r = s(q-1)+t$, gdzie $0 \leq t < q-1$, i rozważmy funkcję $\overline{p}(Z) = p \circ \Phi : \mathbb{F}_{q^m} \to \mathbb{F}_{q^m}$. Potraktujmy ją jako wielomian, wtedy

$$\overline{p}(Z) = (p \circ \Phi)(Z) = p(\Phi_1(Z), \ldots, \Phi_m(Z)) \mod (Z^{q^m} - Z) \ .$$

Z Lematu 10.30 mamy

$$\deg_q(\overline{p}) \leq r$$

Z drugiej strony, z konstrukcji wynika, że

$$\deg(\overline{p}) < q^m \ .$$

W takim razie z Lematu 10.29(4) mamy

$$\deg(\overline{p}) \leq q^m(q-t)q^{m-s-1} \ ,$$

co kończy dowód. □

Rozdział 11

Kody cykliczne

Na podstawie [4, Rozdział 7]

Definicja 11.1. Podzbiór $S \subseteq \mathbb{F}_q^n$ jest *cykliczny*, jeśli $(a_1, \ldots, a_{n-1}, a_n) \in S \implies (a_n, a_1, \ldots, a_{n-1}) \in S$.

Kod liniowy C nazywany jest *kodem cyklicznym*, jeżeli C jest zbiorem cyklicznym.

Łatwo sprawdzić, że kod dualny do kodu cyklicznego też jest cykliczny (ćwiczenie).

Przykład 11.2. Kod RS ewaluowany w każdym punkcie poza 0 i kolejnością punktów ewaluacji $\gamma^0, \ldots, \gamma^{n-1}$ jest kodem cyklicznym.

Kod RS ewaluowany w każdym punkcie, łącznie z 0, *nie jest* kodem cyklicznym.

Przykład 11.3. Poniższe kody są kodami cyklicznymi:

- kod binarny $[3, 2, 2]$-liniowy $\{000, 110, 101, 011\}$;
- kod simplex $(3, 2) = \{0000000, 1011100, 0101110, 0010111, 1110010, 0111001, 1001011, 1100101\}$ (czyli kod dualny do kou Hamminga).

Uwaga. Cykliczność to dość skomplikowana cecha:

- Kody cykliczne *nie są zamknięte* na permutację współrzędnych.
- Kody cykliczne *nie są zamknięte* na usunięcie współrzędnej.

Cykliczność jest własnością kombinatoryczną, przekształcimy ją we własność algebraiczną, używając następującej odpowiedniości:

$$\pi : \mathbb{F}_q^n \to \mathbb{F}_q[X]/(X^n - 1), \pi(a_0, \ldots, a_{n-1}) = a_0 + a_1 X + \cdots + a_{n-1} X^{n-1} \ .$$

Zauważmy, że π jest bijekcją liniową pomiędzy dwoma przestrzeniami liniowymi nad $\mathbb{F}_q$. W związku z tym identyfikujemy $\mathbb{F}_q^n$ z $\mathbb{F}_q[X]/(X^n - 1)$, a wektor $\vec{u} = (u_0, \ldots, u_{n-1})$ z wielomianem $\vec{u}(X) = \sum_{i=0}^{n-1} u_i X^i$. Wiemy, że $\mathbb{F}_q[X]/(X^n - 1)$ jest pierścieniem, nie jest ciałem (poza trywialnym przypadkiem $n = 1$). Ale w ten sposób mamy zdefiniowaliśmy operacją mnożenia elementów w $\mathbb{F}_q^n$.

Fakt 11.4. *Przesunięcie cykliczne (o jedną pozycję) w $\mathbb{F}_q^n$ odpowiada przemnożenie przez X w $\mathbb{F}_q[X]/(X^n - 1)$.*

Przykład 11.5. Dla kodu cyklicznego $C = \{000, 110, 101, 011\}$ mamy

$$\pi(C) = \{0, 1 + X, 1 + X^2, X + X^2\} \subset \mathbb{F}_2[X]/(X^3 - 1) \ .$$

Przypomnijmy definicję ideału:

Definicja 11.6. Niech R będzie pierścieniem. Niepusty podzbiór $\emptyset \neq I \subseteq R$ nazywany *ideałem*, jeśli

- $a, b, \in I \implies a + b \in I$ dla wszystkich $a, b \in I$;
- $a \in I \implies ba \in I$, dla $b \in R$ oraz $a \in I$.

Przykład 11.7. W pierścieniu $\mathbb{F}_2[X]/(X^3 - 1)$, podzbiór $I = \{0, 1 + X, X + X^2, 1 + X^2\}$ jest ideałem.

W pierścieniu $\mathbb{F}_q[X]/(X^n - 1)$, dla dzielnika $g(X)|(X^n - 1)$, zbiór wielomianów podzielnych przez $g(X)$ tworzą ideał.

Definicja 11.8. Ideał I pierścienia R nazywany jest ideałem głównym, jeżeli istnieje element $g \in I$ taki, że $I = \{ag \ : \ a \in R\} = \langle g \rangle$.

Taki element g nazywamy *generatorem* I; równoważnie, I jest generowany przez g.

Pierścień R jest pierścieniem ideałów głównych, jeśli każdy ideał R jest główny.

Generator może nie być jedyny, mogą też być zbiory generatorów różnej mocy.

Przykład 11.9. W Przykładzie 11.7 ideał I jest główny: $I = \langle 1 + X \rangle$.

Twierdzenie 11.10. *Pierścienie $\mathbb{Z}$, $\mathbb{F}_q[X]$ i $\mathbb{F}_q[X]/(X^n - 1)$ są pierścieniami ideałów głównych*

Dowód. Zauważmy, że jeśli $a, b \in I$ to też $\text{nwd}(a, b) \in I$ (o ile jest zdefiniowane i wyraża się jako $aa' + bb'$) oraz $a, b \in \langle \text{nwd}(a, b) \rangle$. Wtedy każdy ideał jest ideałem głównym: ze zbioru generatorów można wziąć dowolne dwa i zastąpić przez jeden (przy nieskończonych trzeba coś jeszcze uargumentować, ale to nas nie dotyczy).

Wiemy, że nwd ma taką własność w $\mathbb{Z}$ oraz w $\mathbb{F}_q[X]$.

Ale dla wielomianów modulo $X^n - 1$ można policzyć nwd bez modulo i on też będzie się wyrażał jako kombinacja po zaaplikowaniu modulo. Ma też mniejszy stopień, czyli wszystko działa. □

11.1 Wielomiany generujący

Twierdzenie 11.11. *Niepusty podzbiór $\emptyset \subseteq C \subseteq \mathbb{F}_q^n$ jest kodem cyklicznym wtedy i tylko wtedy, gdy $\pi(C)$ jest ideałem $\mathbb{F}_q[X]/(X^n - 1)$.*

Dowód. $\Leftarrow$

Pokażmy najpierw, że C jest kodem liniowym. Zauważmy, że π jest przekształceniem liniowym, to wystarczy pokazać, że $\pi(C)$ jest podprzestrzenią liniową. Ale warunek bycia ideałem narzuca też bycie podprzestrzenią liniową (dodawanie, mnożenie przez skalar = mnożenie przez element αX^0). Czyli $\pi(C)$ jest kodem i tym samym C jest kodem.

Chcemy pokazać, że C jest kodem cyklicznym. Niech $(a_0, \ldots, a_{n-1}) \in C$, czyli $\sum_{i=0}^{n-1} a_i X^i \in \pi(C)$. Jako że $\pi(C)$ jest ideałem, to również $X \sum_{i=0}^{n-1} a_i X^i \in \pi(C)$:

$$\begin{aligned} X \sum_{i=0}^{n-1} a_i X^i &= \sum_{i=0}^{n-1} a_i X^{i+1} \\ &= \sum_{i=0}^{n-2} a_i X^{i+1} a_{n-1} X^n \\ &\equiv_{X^n-1} a_{n-1} X^0 + \sum_{i=0}^{n-2} a_i X^{i+1} \end{aligned}$$

co pokazuje, że $(a_{n-1}, a_0, \ldots, a_{n-2}) \in C$, czyli C jest cykliczny.

$\Leftarrow$ Skoro π jest przekształceniem liniowym oraz C jest kodem liniowym, to $\pi(C)$ jest podprzestrzenią liniową. Czyli jeśli $a, b \in \pi(C)$ to też $a + b \in \pi(C)$. Co więcej, z tego, że jest to przestrzeń liniowa, wystarczy sprawdzić dla jednomianów X^d, że jeśli $a \in \pi(C)$ to też $X^d a \in \pi(C)$. Ale do tego starczy tylko dla $d = 1$ i przez indukcję. □

Twierdzenie 11.12. *Niech I będzie niezerowym ideałem w $\mathbb{F}_q[X]/(X^n - 1)$ i niech $g(X)$ będzie niezerowym wielomianem najmniejszego stopnia w I. Wtedy $g(X)$ jest generatorem I oraz $g(X)|(X^n - 1)$. Jest jedyny taki moniczny g.*

W drugą stronę, jeśli moniczny $g|X^n - 1$ to ideał geenrowany przez g potratkowany jako kod jest też generowany przz g.

Dowód. Pierwsza część jest podobnie jak poprzednio: gdyby nie był generatorem, to nwd z innym wielomianem dałoby wielomian mniejszego stopnia w ideale.

Rozważmy dzielenie

$$X^n - 1 = s(X)g(X) + r(X)$$

Wtedy $\deg(r(X)) < \deg(g(X))$. W związku z tym $r(X) = (X^n - 1) - s(X)g(X) \equiv -s(X)g(X)$ jest w ideale. Czyli jest równy 0, z wyboru g.

Jeśli $g \neq g'$ są takie jak w sformułowaniu, to $g - g' \in I$, co jest sprzecznością, bo jest niższego stopnia.

Wiemy, że ideał generowany przez g jako kod ma też generator, ale może to być inny wielomian (np. jeśli $\text{nwd}(g, X^n - 1) = 1$ to ideał generowany przez g to cały pierścień, w takim razie geenrator w sensie kodu to 1). Niech r bedzie generatorem kodu, w szczególności należy do ideału, czyli dla pewnych $h, q \in c\mathbb{F}[X]$ zachodzi równość

$$gh = r + q(X^n - 1) \ .$$

Jako że g dzieli lewą stronę, to dzieli też prawą, a skoro dzieli $X^n - 1$ to dzieli też r, czyli $\deg(r) \geq \deg(g)$, co kończy dowód. □

Definicja 11.13. Jedyny niezerowy wielomian o wiodącym współczynniku 1, najmniejszym stopniu w ideale $I \subseteq \mathbb{F}_q[X]/(X^n - 1)$ nazywany *wielomianem generującym I*. Dla kodu cyklicznego C, wielomian generujący $\pi(C)$ jest też nazywany wielomianem generującym C.

Fakt 11.14. *Każdy wielomian $(X - \alpha)|(X^n - 1)$ jest wielomianem generującym kod cykliczny. W szczególności, jest bijekcja między nimi.*

Przykład 11.15.
- Wielomian generujący kod cykliczny $\{000, 110, 011, 101\}$ to $1 + X$.
- Wielomian generujący kodu simpleks z Przykładu 11.3 to $1 + X^2 + X^3 + X^4$.

Twierdzenie 11.16. *Niech $X^n - 1 \in \mathbb{F}_q[X]$ ma faktoryzację*

$$X^n - 1 = \prod_{i=1}^{r} p_i^{r_i} \ ,$$

gdzie $p_1, \ldots, p_r$ to wszystkie nierozkładalne moniczne dzielniki tego wielomianu. Wtedy liczba kodów cyklicznych długości n nad $\mathbb{F}_q$ wynosi

$$\prod_{i=1}^{r} (e_i + 1) \ .$$

.

Twierdzenie 11.17. *Niech $g(X)$ będzie wielomianem generującym ideał w $\mathbb{F}_q[X]/(X^n-1)$. Jeśli $\deg(g(X)) = n-k$ to odpowiadający kod cykliczny ma wymiar k.*

Dowód. Zauważmy, że dla jednomianów $X^0, \ldots, X^{k-1}$ odpowiadające wielomiany $gX^0, \ldots, gX^{k-1}$ są w ideale oraz są różne (bo mają stopień mniejszy niż n). Tak więc wymiar kodu wynosi przynajmniej k. Gdyby wynosił więcej, to można by rozszerzyć powyższy zestaw o wektor niezależny f, bzo. o taki o najmniejszym stopniu. Musiałby on być wyższego stopnia niż $k-1$, ale wtedy gf się redukuje do mniejszego stopnia. □

11.2 Macierz generatorów i macierz parzystości

11.2.1 Macierz generatorów

Zajmiemy się obecnie macierzami generatorów i macierzami kontroli parzystości dla kodów cyklicznych. Jak łatwo się domyślić, skoro kod jest całkowicie zdeterminowany przez swój wielomian generujący, to również macierz generatorów oraz macierz parzystości można (łatwo) wyliczyć na podstawie wielomianu generującego.

Twierdzenie 11.18. *Niech $g(X) = \sum_{i=0}^{n-k} g_i X^i$ będzie wielomianem generującym kodu cyklicznego $C \subseteq \mathbb{F}_q^n$ z $\deg(g(X)) = n-k$. Wtedy macierz*

$$G_C = [g(X)\, Xg(X)\, \cdots\, X^{k-1}g(X)] = \begin{bmatrix} g_0 & 0 & 0 & \cdots & 0 \\ g_1 & g_0 & 0 & \cdots & 0 \\ g_2 & g_1 & g_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{n-k} & g_{n-k-1} & g_{n-k-2} & \cdots & 0 \\ 0 & g_{n-k} & g_{n-k-1} & \cdots & 0 \\ 0 & 0 & g_{n-k} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & g_{n-k} \end{bmatrix}$$

jest macierzą generująca C.

Dowód. Wystarczy pokazać, że $g(X)$, $Xg(X)$, …, $X^{k-1}g(X)$ są bazą C. Oczywiście są one liniowo niezależne nad $\mathbb{F}_q$. Z drugiej strony wiemy już, że wymiar $\dim(C) = k$, co daje tezę. □

Uwaga. Zauważmy, że dla np. kodów RS to daje istotnie inną reprezentację, niż tę, do której jesteśmy przyzwyczajeni. Ale można pomnożyć (powiedzmy z lewej) przez macierz Vandermonde'a (czyli odwracalną), bo dostać taką reprezentację, jaką byśmy chcieli.

11.2.2 Macierz parzystości

Macierz parzystości można uzyskać w standardowy sposób, albo przez skorzystanie z charakterystyki kodu dualnego.

Okazuje się jednak, że można to scharakteryzować jeszcze prościej. Spróbujmy najpierw przyjrzeć się idei.

Wiemy, że $g | X^n - 1$, niech zatem h będzie taki, że $gh = X^n - 1$, zapiszmy $g(X) = \sum_{i=0}^{n-1} g_i X^i$ oraz $h(X) = \sum_{i=0}^{n-1} h_i X^i$ (wiele z tych współczynników jest zerowych) Wtedy

$$\begin{aligned} X^n - 1 &= gh \\ &= \sum_{k=0}^{2n-2} \left(\sum_{i,j:i+j=k} g_i h_j \right) X^k \end{aligned}$$

Jeśli weźmiemy teraz modulo $X^n - 1$ to dostaniemy

$$\begin{aligned} 0 &\equiv \sum_{k=0}^{2n-2} \left(\sum_{i,j:i+j=k} g_i h_j \right) X^k \\ &\equiv \sum_{k=0}^{n-1} \underbrace{\left(\sum_{i,j:i+j\in\{k,n+k\}} g_i h_j \right)}_{=0} X^k \end{aligned}$$

Wielomian po prawej stronie przystaje modulo $X^n - 1$ do 0, czyli musi mieć wszystkie współczynniki równe 0, bo tam są wykładniki mniejsze niż n. Czyli dostajemy, że dla każdego $k \in \{0, \ldots, n-1\}$

$$0 = \sum_{i,j:i+j\in\{k,n+k\}} g_i h_j$$

Łatwo zauważyć, że dla kolejnych k to są iloczyny skalarne wektora $(g_0, \ldots, g_{n-1})$ z przesunięciami cyklicznymi wektora $(h_{n-1}, \ldots, h_0)$. Trzeba to sformalizować i pokazać, że wśród nich jest baza. Łatwo też zobaczyć, że h prawie jest poszukiwanym wielomianem generującym dla kodu dualnego — trzeba tylko wypisać współczynniki w odwrotnej kolejności.

Definicja 11.19 (Wielomian odwrotny)**.** Niech $h(X) = \sum_{i=0}^{k} h_i X^i$ będzie wielomianem stopnia k. Wtedy $h_R(X) = X^k h(1/X) = \sum_{i=0}^{k} a_{k-i} X^i$ nazywamy *wielomianem odwrotnym* do h.

Fakt 11.20. *Jeśli $h|X^n - 1$ to $h_R|X^n - 1$*

Prosty dowód pozostawiamy jako ćwiczenie.

Twierdzenie 11.21. *Niech $g(X)$ będzie wielomianem generującym $[n, k]$ kodu cyklicznego C. Zdefiniujmy $h(X) = (X^n - 1)/g(X)$. Wtedy $h_0^{-1} h_R(X)$ jest wielomianem generującym dualny kod cykliczny $C^\perp$, gdzie h_0 jest stałą wielomianu $h(X)$.*

Dowód. Dowód wynika z rachunków powyżej, mnożnik h_0^{-1} jest tylko po to, by unormować wielomian (czynnik wiodący h_R to h_0).

Łatwo sprawdzić, że $h_0 \neq 0$: gdyby tak było, to wtedy $X|h(X)$ i tym samym $X|X^n - 1$, co nie jest prawdą. □

Definicja 11.22. Przy założeniach Twierdzenia 11.21 wielomian $h_0^{-1} h_R(X)$ nazywamy wielomianem (kontroli) parzystości kodu C.

Wniosek 11.23. Niech C będzie $[n, k]_q$ kodem cyklicznym z wielomianem generującym $g(X)$. Niech $h(X) = (X^n - 1)/g(X)$ i $h(X) = \sum_{i=0}^{k} h_i X^i$ Wtedy macierz

$$\begin{bmatrix} h_R(X) \\ X h_R(X) \\ \vdots \\ X^{n-k-1} h_R(X) \end{bmatrix} = \begin{bmatrix} h_k & h_{k-1} & \cdots & h_0 & 0 & \ldots & 0 \\ 0 & h_k & \cdots & h_1 & h_0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & h_k & h_{k-1} & \cdots & h_0 \end{bmatrix}$$

jest macierzą parzystości kodu C.

Z tej postaci łatwo jest też otrzymać macierz parzystości postaci

$$[I|H'] \ .$$

11.3 Kody Golay'a

Na podstawie [3, Wykład 7] oraz [4, Rozdział 5.3.3].

Rozważmy pierścieniu $\mathbb{Z}_2[X]$ i wielomian:

$$X^{23}-1 = (X-1)\cdot(X^{11}+X^{10}+X^6+X^5+X^4+X^2+1)\cdot(X^{11}+X^9+X^7+X^6+X^5+X+1) \ .$$

Niech

$$w(X) = X^{11}+X^{10}+X^6+X^5+X^4+X^2+1 \ .$$

Wtedy wielomian

$$w_R(X) = X^{11}+X^9+X^7+X^6+X^5+X+1 \ .$$

Wobec tego,

$$X^{23}-1 = (X-1)\cdot w(X)\cdot w_R(X) \ .$$

Definicja 11.24 (Kod Golay'a)**.** (Binarnym) *kodem Golay'a* nazywamy kod cykliczny zawarty w przestrzeni $\mathbb{F}_2[X]/(X^{23}-1)$ generowany przez wielomian

$$w(X) = X^{11}+X^{10}+X^6+X^5+X^4+X^2+1 \ .$$

Rozszerzony kod Golay'a definiowany jest w naturalny sposób, jest to $[24, 12]$ kod.

Twierdzenie 11.25. *Kod Golaya jest* $[23, 12, 7]$*-kodem doskonałym.*

Rozszerzony kod Golay'a jest $[24, 12, 8]$ *kodem. Jest samo-dualny.*

Kod Golay'a ma wiele różnych interpretacji kombinatorycznych.

Kod Golaya można rozłożyć nad $\mathbb{F}_{2048}$. Co ciekawe, pierwiastki można podać dość łatwo:

Fakt 11.26. *Jeśli* $w \in \mathbb{F}_2[X]$ *ma pierwiastek* $a \in \mathbb{F}_{2^k}$*, to* a^2 *też jest pierwiastkiem.*

Fakt 11.27. *Ciało* $\mathbb{F}_{2048}$ *ma element rzędu* 23*.*

Fakt 11.28. *Dla pewnego* $p \in \mathbb{F}_{2048}$ *rzędu* 23 *wielomian* w *ma pierwiastki* p, p^2, p^4, p^8, p^{16}*,* p^9*,* p^{18}*,* p^{13}*,* p^3, p^6, p^{12}*.*

Tym samym wielomian w_R *ma pierwiastki* $p^5, p^7, p^{10}, p^{11}, p^{14}, p^{15}, p^{17}, p^{19}, p^{20}, p^{21}, p^{22}$*.*

11.4 Korekcja błędów dla kodów cyklicznych

Twierdzenie 11.29. *Niech* H *będzie macierzą parzystości wymiaru* $(n-k)\times n$ *postaci* $(\mathrm{Id}\,|A)$ *dla kodu cyklicznego* C *generowanego przez* $g(X)$*. Wtedy syndrom* Hw *wektora* $w \in \mathbb{F}_q^n$ *jest równy* $w(X) \bmod g(X)$*.*

Dowód. Potraktujmy kolumny macierzy A jako wielomiany stopnia (stopnia $< n-k$), oznaczmy je przez $a_0, \ldots, a_{k-1}$, tak że

$$A = (a_0(X)|a_1(X)|\cdots|a_{k-1}(X)) \ .$$

Wiemy (a w razie czego można łatwo sprawdzić), że jedna z macierzy generujących C jest postaci

$$G = (-A|I)$$

Dlatego też $X^{n-k+i} - a_i(X)$ jest słowem kodowym C. Niech $q_i \in \mathbb{F}_q[X]/(X^n-1)$ będzie takim wielomianem, że

$$X^{n-k+i} - a_i(X) = q_i(X)g(X)$$

Niech

$$w(X) = \sum_{i=0}^{n-1} w_i X^i \ .$$

Wtedy syndrom wynosi

$$s = Hw$$

a odpowiadający wielomian $s(X)$ jest równy

$$\begin{aligned} s(X) &= \underbrace{\sum_{i=0}^{n-k-1} w_i X^i}_{\text{z macierzy Id}} + \underbrace{\sum_{j=0}^{k-1} w_{n-k+j} a_j(X)}_{\text{z macierzy } A} \\ &= \sum_{i=0}^{n-k-1} w_i X^i + \sum_{j=0}^{k-1} w_{n-k+j} \left(X^{n-k+j} - q_j(X)g(X) \right) \\ &= \underbrace{\sum_{i=0}^{n-1} w_i X^i}_{w(X)} - g(X) \sum_{j=0}^{k-1} w_{n-k+j} q_j(X) \\ &\equiv w(X) \mod g(X) \ . \end{aligned}$$

Ponieważ $\deg(s) \leq n-k-1$, otrzymujemy, że jest on równy reszcie z dzielenia w przez g. □

Uwaga. Syndrom względem innej macierzy parzystości może być inny (jest przekształcony przez jakąś macierz odwracalną). Niektórzy używają w odniesieniu do kodów cyklicznych słowa *syndrom* po prostu na określenie reszty z dzielenia wielomianu wiadomości przez wielomian generujący i nie wnikają, jak się go liczy i jaki ma związek z macierzą parzystości.

Wniosek 11.30. Niech $g(X)$ będzie wielomianem generującym kodu cyklicznego C. Jeśli dla otrzymanej wiadomości $w(X)$ mamy, że reszta $s(X) = w(X) \bmod g(X)$ ma wagę Hamminga $\mathrm{d_H}(s(X)) < \frac{\mathrm{d_H}(C)-1}{2}$, to $s(X)$ jest wektorem błędu dla $w(X)$, tj. $w(X) - s(X)$ jest najbardziej prawdopodobnym słowem kodowym.

11.4.1 Error trapping

Nie działa zawsze, działa jeśli jest dostatecznie długi ciąg bez błędów.

Lemat 11.31. *Niech C będzie $[n,k]_q$ kodem cyklicznym generowanym przez wielomian $g(X)$. Niech $s(X) = \sum_{i=0} s_i X^i$ będzie wielomianem syndromu $w(X)$. Wtedy syndrom przesunięcia cyklicznego $Xw(X)$ jest równy $Xs(X) - s_{n-k-1}g(X)$.*

Dowód. Z Twierdzenia 11.29 wystarczy pokazać, że $Xs(X) - s_{n-k-1}g(X)$ to reszta z dzielenia $Xw(X)$ przez $g(X)$.

Niech $w(X) = q(X)g(X) + s(X)$. Wtedy

$$Xw(X) = Xq(X)g(X) + Xs(X)$$

Zauważmy, że $Xq(X)$ oraz $Xs(X)$ spełniaja odpowiednie zależności, ale stopień s może być za duży, co oznacza, że s_{n-k-1} jst niezerowe i teraz jest przy potędze X^{n-k}

$$Xw(X) = (Xq(X) + s_{n-k-1})g(X) + (Xs(X) - s_{n-k-1}g(X)) \ .$$

Tezę dostajemy z obserwacji, że $\deg(Xs(X) - s_{n-k-1}g(X)) < n-k = \deg(g(X))$. □

Zauważmy, że w ten sposób możemy policzyć (w czasie kwadratowym) syndromy wszystkich przesunięć cyklicznych $w(X), Xw(X), X^2, \ldots X^{n-1}w(X)$.

Definicja 11.32. Cykliczny powtórzenie 0 długości ℓ w n-krotce to ciąg kolejnych (wg. przesunięcia cyklicznego) ℓ zer w tej krotce.

Algorytm dekodowania dla kodów cyklicznych

Algorytm 3 Algorytm Error trapping

Założenie: C jest $[n,k,d]_q$-kodem cyklicznym generowanym przez wielomian $g(X)$. $w(X)$ jest odebranym komunikatem, zaś $e(X)$ będzie błędem, przy czym $\|e(X)\|_1 \leq \frac{d-1}{2}$ oraz ma cykliczne powtórzenie 0 długości przynajmniej k.

$s(X) \leftarrow w(X) \mod g(X)$ ▷ $\mathcal{O}(n^2)$
if $\|s(X)\|_1 \leq \frac{d-1}{2}$ **then**
  return $w(X) - s(X)$
for $i \leftarrow 1 \,..\, n-1$ **do**
  $s(X) \leftarrow Xs(X) - s_{n-k-1}g(X)$ ▷ $\mathcal{O}(n)$
  if $\|s\|_1 \leq \frac{d-1}{2}$ **then**
    $e(X) \leftarrow X^{n-i}s(X) \bmod (X^n - 1)$
    return $w(X) - e(X)$

Twierdzenie 11.33. *Niech C będzie $[n,k,d]_q$-kodem cyklicznym generowanym przez wielomian $g(X)$. Niech $w(X)$ będzie odebranym komunikatem, zaś $e(X)$ będzie błędem, przy czym $\|e(X)\|_1 \leq \frac{d-1}{2}$ oraz ma cykliczne powtórzenie 0 długości przynajmniej k. Wtedy algorytm Error Trapping poprawnie dekoduje wiadomość.*

Uwaga, takie m nie jest jedyne, ale z wcześniejszej uwagi, jeśli algorytm coś zwróci, to wynik jest poprawny.

Dowód. Przypomnijmy, że stopień $\deg(g) = n-k$. Wystarczy pokazać, że takie m istnieje, przy założeniu, że istnieje wektor błędu $e(X)$ taki, że $e(X)$ ma podciąg cykliczny 0 o długości co najmniej k. Pozostałe kroki oraz ich poprawność są wtedy jasne.

Weźmy m takie, że przesunięcie cykliczne $e(X)$ o m pozycji sprawia, że wszystkie niezerowe pozycje $e(X)$ będą w obrębie pierwszych $n-k$ pozycji, czyli jest to wielomian stopnia $< n-k$. Dla prostoty prezentacji przyjmijmy, że $m = 0$. Wtedy $\deg(e(X)) < n-k = \deg(g(X))$. Ale w takim razie $w(X) \bmod g(X) = e(X)$. Oczywiście $\|e(X)\|_1 \leq \frac{d-1}{2}$, co oznacza, że istnieje takie m. To, że zwraca poprawną wartość, jest jasne z kodu (cofa przesuniecie cykliczne). □

11.5 Poprawianie błędów pęknięć (burst)

Będziemy się zajmować problemem poprawiania błędów, które są „w seriach". Jednym ze sposobów jest użycie większego ciała i użycie kodów RS, ale można też bezpośrednio.

Definicja 11.34. Pęknięciem (burst) długości $\ell > 1$ nazywamy wektor binarnym, którego niezerowe współrzędne zawarte są w ℓ cyklicznie kolejnych pozycjach. Przy pęknięciu niecyklicznym patrzymy na pozycje kolejne, nie na kolejne cyklicznie.

Kod nazywany jest kodem korygującym ℓ błędy pęknięć, jeśli potrafi skorygować wszystkie pęknięcia długości najwyżej ℓ.

Przykład 11.35. `001101000000` jest pęknięciem o długości 4, zaś `0100000000000000100` to pęknięcie o długości 5.

Lemat 11.36. *Kod liniowy C jest kodem korygującym błędy ℓ-pęknięć wtedy i tylko wtedy, gdy jeśli wszystkie pęknięcia o długości najwyżej ℓ są w różnych warstwach C.*

Dowód. Jeżeli wszystkie pęknięcia długości najwyżej ℓ leżą w różnych warstwach, to każdy z błędów pęknięć jest jednoznacznie określany przez jego syndrom i można go jednoznacznie skorygować.

Z drugiej strony, załóżmy, że dwa różne błędy pęknięć e_1 i e_2 długości najwyżej ℓ leżą w tej samej warstwie C. Wtedy $c = e_1 - e_2$ jest słowem kodowym. Tak więc odbierając e_1 możemy zdekodować zarówno na $\vec{0}$ (z błędem e_1) jak i $\vec{c}$ (z błędem e_2). □

Wniosek 11.37. Niech C będzie $[n,k]_q$ kodem korygującym ℓ błędy pęknięć. Wtedy

- Suma dwóch pęknięć długości najwyżej ℓ nie może być słowem kodowym.
- żadne niezerowe pęknięcie o długości najwyżej 2ℓ nie może być słowem kodowym;
- (ograniczenie Reiger'a) $n - k \geq 2\ell$

Dowód. Z powyższego Lematu wynika część pierwsza (jeśli różnica jest słowem kodowym, to są w jednej warstwie.)

W szczególności, pęknięcie długości $\leq 2\ell$ jest sumą dwóch pęknięć długości $\leq \ell$.

Niech $u_1, \ldots, u_{n-k+1}$ to pierwszych $n-k+1$ wektorów kolumnowych macierzy kontroli parzystości dla kodu C.

Jako że są one w $\mathbb{F}^{n-k}$ to są one liniowo zależne, czyli są stałe $c_1, \ldots, c_{n-k+1} \in \mathbb{F}_q$, nie wszystkie zerowe, takie że

$$\sum_{i=1}^{n-k+1} c_i \vec{u_i} = \vec{0} \ .$$

Oznacza to, że $(c_1, \ldots, c_{n-k+1}, \vec{0})$ jest słowem kodowym, oraz oczywiście jest to pęknięciem o długości $\leq n-k+1$. Z pokazanej części pierwszej dostajemy, że $n-k+1 > 2\ell$, tj. $n-k \geq 2\ell$, co daje tezę. □

Liniowy kod korygujący błędy pęknięć osiągający powyższe ograniczenie Reigera nazywany *optymalnym kodem korygującym błędy.*

Uwaga. Zauważmy, że kody spełniające ograniczenie Reiger'a jest „słabsze", niż ograniczenie Singleton'a: dla kodow spełniających ograniczenie Singletona z rownością umiemy poprawić

$$\frac{d-1}{2} = \frac{n-k}{2}$$

błędów. A dla kodów optymalnych potrafimy poprawić błędy pęknięć długości

$$\ell = \frac{n-k}{2}$$

Czyli taka sama liczba, ale muszą być zawarte w segmencie.

Ale kody MDS binarne są tylko trywialne, zaś optymalnych jest sporo.

Zauważmy, że pęknięcie długości ℓ ma cykliczne powtórzenie 0 długości $n - \ell$. Z Wniosku 11.37 mamy

$$k \leq n - 2\ell < n - l$$

dla $[n,k]$ kodu liniowego korygującego ℓ-błędy pęknięć. Spełnia to wymóg algorytmu error trapping z poprzedniego rozdziału. Możemy go więc bezpośrednio zastosować do korekcji błędów pęknięć; jedyna różnica, to że w przypadku korekty błędów pęknięć, nie mamy założenie na wagę błędu (że jest mniejsza niż $\frac{\mathrm{d_H}(C)-1}{2}$, a zamiast tego zakładamy, że błąd jest pęknięciem długości najwyżej ℓ.

Algorytm 4 Algorytm Error trapping wersja dla pęknięć

Założenie: C jest $[n, k, d]_q$-kodem cyklicznym generowanym przez wielomian $g(X)$. $w(X)$ jest odebranym komunikatem, zaś $e(X)$ będzie błędem i jest pęknięciem długości najwyżej ℓ.

$s(X) \leftarrow w(X) \mod g(X)$ ▷ $\mathcal{O}(n^2)$

if s jest niecyklicznym pęknięciem długości najwyżej ℓ **then**

return $w(X) - s(X)$

for $i \leftarrow 1 \,..\, n-1$ **do**

$s(X) \leftarrow Xs(X) - s_{n-k-1}g(X)$ ▷ $\mathcal{O}(n)$

if s jest niecyklicznym pęknięciem długości najwyżej ℓ **then**

$e(X) \leftarrow X^{n-i}s(X) \bmod (X^n - 1)$

return $w(X) - e(X)$

Parametry kodu	Wielomian generujący
$[7, 3]$	$1 + X + X^3 + X^4$
$[15, 9]$	$1 + X + X^2 + X^3 + X^6$
$[15, 7]$	$1 + X^4 + X^6 + X^7 + X^8$
$[15, 5]$	$1 + X + X^2 + X^4 + X^5 + X^8 + X^{10}$

Tablica 11.1: Kilka optymalnych kodów korygujących błędy pęknięć

Rozdział 12

Kody BCH

Głównie na podstawie [4, Rozdział 8], w małym stopniu [9, Rozdział 6] oraz [8, Zadania do rozdziału 5].

Kody BCH (*B*ose, Ray-*C*haudhuri i *H*ocquenghem) zdefiniowali niezależnie A. Hocquenghem [8] w 1959 oraz R. C. Bose i D. K. Ray-Chaudhuri w 1960 roku. Uogólnienie binarnych kodów BCH do kodów q-arnych zaproponował D. Gorensteina i N. Zierlera [5] w 1961 roku.

Można myśleć, że jest to uogólnienie kodów Hamming'a do korekcji wielu błędów. Albo że są to kody zawarte w kodach RS.

12.1 Przykład wprowadzający

Przykład 12.1. Niech $\alpha \in \mathbb{F}_{q^m}$ będzie elementem pierwotnym (czyli generatorem $\mathbb{F}^*_{q^m}$), niech $M^{(i)}(X)$ będzie minimalnym wielomianem dla α^i nad $\mathbb{F}_q$. Wtedy również każdy inny pierwiastek β wielomianu $M^{(i)}(X)$ jest elementem $\mathbb{F}_{q^m}$, czyli β spełnia $\beta^{q^m-1} - 1 = 0$; innymi słowy, $X - \beta$ jest liniowym dzielnikiem $X^{q^m-1} - 1$. Wielomian minimalny nie wielokrotnych pierwiastków, w takim razie $M^{(i)}(X) | X^{q^m-1} - 1$. Co więcej, dla zbioru tego typu wielomianów $\{M^{(i)}(X)\}_{i \in I}$ oczywiście $\mathrm{nww}(\{M^{(i)}(X)\}_{i \in I}) | X^{q^m-1} - 1$.

Powyższy przykład przedstawia sposób na znalezienie niektórych dzielników $X^{q^m-1} - 1$. Dzielniki te mogą być wybrane jako wielomiany generatora cyklicznych kodów długości $q^m - 1$.

12.2 Minimalne wielomiany

Definicja 12.2. Niech $n = q^m - 1$ dla q — potęgi liczby pierwszej. Warstwa q-cyklotomiczna modulo n zawierającą i to

$$C_i = \{i \cdot q^j \mod n \ := 0, 1, \ldots m-1\} \ .$$

Uwaga. Łatwo zauważyć, że:

- C_i, C_j są równe lub rozłączne.
- Co więcej, branie większych j nic nie zmienia.
- $|C_i| \leq m$ i równość zachodzi wtw $\mathrm{nwd}(i, n) = 1$.

Twierdzenie 12.3. *Niech α będzie elementem pierwotnym $\mathbb{F}_{q^m}$. Wtedy minimalny wielomian dla α^i nad $\mathbb{F}_q$ to*

$$M^{(i)}(X) = \prod_{j \in C_i} (X - \alpha^j),$$

gdzie C_i jest cyklotomiczną warstwą modulo $q^m - 1$ zawierającą i.

Dowód. Należy najpierw sprawdzić, że:

1. α^i jest jego pierwiastkiem.

2. jest to wielomian o współczynnikach z $\mathbb{F}_q$;

3. jest nierozkładalny nad $\mathbb{F}_q$;

Część 1 jest jasne, dla $j = 0$.

Część 2: wystarczy pokazać, że jeśli podniesiemy współczynniki do potęgi q, to dostaniemy ten sam wielomian (co oznacza, że te współczynniki są z $\mathbb{F}_q$). Niech

$$M^{(i)}(X) = \sum_{i=0}^{r} a_i X^i \ ,$$

gdzie $a_k \in \mathbb{F}_{q^m}$ i $r = |C_i|$. Wtedy

$$\begin{aligned}\sum_{i=0}^{r} a_i^q X^i &= \prod_{j \in C_i} (X - \alpha^{qj}) \\ &= \prod_{j \in C_i} (X - \alpha^j) \\ &= M^{(i)}(X)\end{aligned}$$

Przy czym pierwsza równość wynika z tego, że a_i jest (z wzorów Vietta) wyraża się jako pewien wielomian od pierwiastków. Jeśli podniesiemy tę równość do q-tej potęgi, to otrzymamy analogiczną zależność na q-te potęgi tych pierwiastków. Np. $a^{r-1} = \sum \alpha_i$, z czego wynika, że $a_i^q = (\sum_i \alpha_i)^q = \sum_i \alpha_i^q$.

Część 3 Jako że α jest elementem pierwotnym, mamy $\alpha^j \neq \alpha^k$ dla dwóch różnych elementów $j, k \in C_i$. Czyli $M^{(i)}(X)$ nie ma pierwiastków wielokrotnych.

Niech $f(X) \in \mathbb{F}_q[X]$ taki że $f(\alpha^i) = 0$. Niech

$$f(X) = \sum_{k=0}^{p} f_k X^k$$

dla pewnych $f_0, \ldots, f_p \in \mathbb{F}_q$. Wystarczy pokazać, że jeśli $f(\alpha^j) = 0$ to $f(\alpha^{jq})$ też jest pierwiastkiem.

$$\begin{aligned}f(\alpha^{jq}) &= \sum_{k=0}^{n} f_k \alpha^{jqk} \\ &= \sum_{k=0}^{n} f_k^q \alpha^{jkq} \\ &= \left(\sum_{k=0}^{n} f_k \alpha^{jk} \right)^q \\ &= f(\alpha^j)^q \\ &= 0\end{aligned}$$

Skoro f dzieli się przez α^i to dzieli się przez wszystkie czynniki liniowe $M^{(i)}$ i tym samym przez samo $M^{(i)}$. Czyli $M^{(i)}$ jest minimalny dla α^i. □

12.3 Najmniejsza wspólna wielokrotność

Niech $\mathrm{nww}(f_1, \ldots, f_k)$ to najmniejsza wspólna wielokrotność wielomianów dla $f_1, \ldots, f_k \in \mathbb{F}[X] \setminus 0$. Ze standardowych argumentów wiemy, że to jest dobrze zdefiniowane. Dla uproszczenia bierzemy wielomian unormowany. Łatwo sprawdzić, że taki wielomian istnieje i że można go policzyć iteracyjnie

$$\mathrm{nww}(f_1, f_2) = \frac{f_1 f_2}{\mathrm{nwd}(f_1, f_2)}$$
$$\mathrm{nww}(f_1, \ldots, f_k) = \mathrm{nww}(f_1, \mathrm{nww}(f_2, \ldots, f_k))$$

Jednocześnie, jeśli rozkłady na czynniki nierozkładalne f_i to

$$f_i = \alpha_i \prod_{j=1}^{\ell} p_j^{e_{i,j}}$$

gdzie $p_1, \ldots, p_\ell$ są nierozkładalne oraz $\alpha_1, \ldots, \alpha_k \in \mathbb{F}^*$, to

$$\mathrm{nww}(f_1, \ldots, f_k) = \prod_{j=1}^{\ell} p_j^{\max_{i=1}^{k} e_{i,j}}$$

Fakt 12.4. *Niech $f_1, \ldots, f_k, g \in \mathbb{F}[X]$; jeśli $f_i | g$ dla $i = 1, \ldots, k$ to $\mathrm{nww}(f_1, \ldots, f_k) | g$.*

12.4 Kody BCH

Definicja 12.5. Niech α będzie elementem pierwotnym $\mathbb{F}_{q^m}$, niech $M^{(i)}(X)$ będzie minimalnym wielomianem α^i nad $\mathbb{F}_q$. Kod BCH nad $\mathbb{F}_q$ o długości $n = q^m - 1$ z założoną odległością δ jest cyklicznym q-arnym kodem generowanym przez $g(X) = \mathrm{nww}(M^{(a)}(X), M^{(a+1)}(X), ., M^{(a+\delta-2)}(X))$ dla pewnej liczby naturalnej a. Ponadto kod jest BCH w sensie ścisłym, jeśli $a = 1$.

Ta definicja daje nam, że kod jest cykliczny (w szczególności: liniowy), ale można też podać ją w inny sposób, bardziej czytelny.

Lemat 12.6. *Kod BCH długośc $n = q^m - 1$ i minimalnej odległości d elemencie pierwotnym $\alpha \in \mathbb{F}_{q^m}$ oraz parametrze startowym a to*

$$\{(c_0, \ldots, c_{n-1}) \ : \ c(\alpha^i) = 0 \ \textit{dla}\ i = a, \ldots, a + d - 2\} \ ,$$

gdzie zgodnie z wcześniejszą konwencją $c(X) = \sum_{i=0}^{n-1} c_i X^i$. Powyższy zbiór jest równy $\mathbb{F}_q^n \cap \ker(H'_C)$, gdzie

$$H'_C = \begin{bmatrix} 1 & \gamma^a & \gamma^{2a} & \ldots & \gamma^{(n-1)a} \\ 1 & \gamma^{a+1} & \gamma^{2(a+1)} & \ldots & \gamma^{(n-1)(a+1)} \\ \vdots & \vdots & \ldots & \vdots & \\ 1 & \gamma^{a+\delta-2} & \gamma^{2(a+\delta-2)} & \ldots & \gamma^{(n-1)(a+\delta-2)} \end{bmatrix}$$

Dowód. Dowód to przepisanie definicji: jeśli coś jest elementem kodu BCH to dzieli się przez każde $M^{(i)}$ dla $i = a, \ldots, a + \delta - 2$ czyli w takim razie zeruje każdy z nich, czyli zeruje się na każdym α^i.

Z drugiej strony, jeśli wielomian c zeruje się na α^i to $M^{(i)}(X) | c(X)$ i jeśli c zeruje się na α^i dla $i = a, \ldots, a + \delta - 2$ to dzieli się przez $M^{(i)}(X)$ dla $i = a, \ldots, a + \delta - 2$ i w takim razie dzieli się też przez ich nww, czyli jest w kodzie BCH.

Warunek z macierzą to tylko przeformułowanie drugiego warunku. □

Ta druga definicja ma pewne zalety: zauważmy, że coś podobnego widzieliśmy już dla kodów RS: w Twierdzeniu 5.9 pokazaliśmy, że dla $n = q^m - 1$ oraz α będącego generatorem $\mathbb{F}_{q^m}^*$. mamy

$$\mathrm{RS}(\alpha^0, \ldots, \alpha^{n-1}, n, k) = \{(c_0, \ldots, c_{n-1}) \,:\, c(\alpha^i) = 0, i = 1, \ldots, d-1 = n-k\} \ ,$$

gdzie $c(X) = \sum_{i-0}^{n-1} c_i X^i$.

Czyli kody BCH dla $a = 1$ to kody RS przycięte do $\mathbb{F}_q$.

Lemat 12.7. *Niech α będzie elementem pierwotnym $\mathbb{F}_{q^m}$. Wtedy ścisły kod BCH nad $\mathbb{F}_q$ o długości $n = q^m - 1$ z minimalną odległością δ oraz $a = 1$ jest równy*

$$\mathrm{RS}(\alpha^0, \ldots, \alpha^{n-1}, n, \delta - 1) \cap \mathbb{F}_q^n \ .$$

W szczególności pokazuje to, że odległość kodów BCH to przynajmniej δ (może być więcej, bo przecinamy kody RS, być może minimalne słowo kodowe wypada).

Ta odległość może być większa.

Przypadek dla ogólnego a pozostawiamy jako ćwiczenie.

Oszacowanie wymiaru kodu również nie jest jednoznaczne.

Przykład 12.8. Niech α będzie elementem pierwotnym $\mathbb{F}_{2^m}$. Wtedy kod binarny BCH w sensie ścisłym o planowanej odległości 2 jest kodem cyklicznym generowanym przez $M^{(1)}(X)$. Jest to w rzeczywistości kod Hamming (co jest zadaniem na poprzedniej liście zadań).

Przykład 12.9. Niech $\alpha \in \mathbb{F}_8$ będzie pierwiastkiem $1 + X + X^3$. Jest to element pierwotny $\mathbb{F}_8$. Wielomiany $M^{(1)}(X)$ i $M^{(2)}(X)$ są równe $1+X+X^3$ (z zadania z listy: jeśli α jest pierwiastkiem, to też α^2 dla ciała charakterystyki 2). Stąd kod binarny BCH o długości 7 w ścisłym sensie generowany przez $\mathrm{nww}(M^{(1)}(X), M^{(2)}(X)) = 1+X+X^3$ to $[7, 4]$-kod. W istocie jest to $[7, 4, 3]$-kod.

12.5 Parametry kodów BCH

Długość kodu BCH oczywiście wynosi $q^m - 1$ (jako kodu cyklicznego), jak już zauważyliśmy, odległości wynosi przynajmniej δ. Oszacujmy wymiar

Twierdzenie 12.10. *1. Wymiar kodu q-arnego kodu BCH o długości $q^m - 1$ generowanego przez $g(X) = \mathrm{nww}(M^{(a)}(X), M^{(a+1)}(X), \ldots, M^{(a+\delta-2)}(X))$ jest niezależny od wybór elementu pierwotnego α.*

2. q-arny kod BCH o długości $q^m - 1$ z założoną odległością δ ma wymiar przynajmniej $q^m - 1 - m(\delta - 1)$.

Dowód. Dowód części pierwszej pozostawimy jako proste ćwiczenie.

Wymiar k spełnia następującą zależność:

$$\begin{aligned} k &= q^m - 1 - |S| \\ &= q^m - 1 - \left| \bigcup_{i=a}^{a+\delta-2} C_i \right| \\ &\geq q^m - 1 - \sum_{i=a}^{a+\delta-2} |C_i| \\ &\geq q^m - 1 - \sum_{i=a}^{a+\delta-2} m \\ &= q^m - 1 - (\delta - 1)m \ , \end{aligned}$$

co kończy dowód. □

Powyższy wynik pokazuje, że aby znaleźć wymiar q-arnego kodu BCH długości $q^m - 1$ generowanego przez $g(X) = \mathrm{nww}(M^{(a)}, \ldots, M^{a+\delta-2})$ wystarczy określić moc $\bigcup_{i=a}^{a+\delta-2} C_i$, gdzie C_i jest warstwą cyklotomiczną modulo $q^m - 1$ zawierającą i.

Jest wiele przykładów, że to oszacowanie nie jest ścisłe. Lecz są też takie, że jest ścisłe, nawet dla całych klas.

Twierdzenie 12.11. *q-arny ścisły kod BCH o długości $q^m - 1$ z założoną odległością δ ma wymiar dokładnie $q^m - 1 - m(\delta - 1)$ jeżeli $q \neq 2$ i $\mathrm{nwd}(q^m - 1, e) = 1$ dla wszystkich $1 \leq e \leq \delta - 1$.*

Dowód. Z dowodu Twierdzenia 12.10 wiemy, że wymiar jest równy

$$q^m - 1 - \left| \bigcup_{i=1}^{\delta-1} C_i \right|$$

gdzie C_i to cyklotomiczna warstwa modulo $q^m - 1$ zawierająca i. W związku z tym wystarczy i należy udowodnić, że $|C_i| = m$ dla wszystkich $1 \leq i \leq \delta - 1$, oraz że C_i i C_j są rozłączone dla wszystkich $1 \leq i < j \leq \delta - 1$.

Po pierwsze, zauważmy, że dla każdej liczby całkowitej $1 \leq t \leq m - 1$ oraz każdej $1 \leq i \leq \delta - 1$ zachodzi $i \not\equiv_{q^m-1} iq^t$. W przeciwnym razie mielibyśmy

$$(q^t - 1)i \equiv_{q^m-1} 0$$

Za założenia mamy $\mathrm{nwd}(i, q^m - 1) = 1$ i tym samym $q^t - 1 \equiv_{q^m-1} 0$, co nie jest możliwe dla $t < m$. Co pokazuje, że $|C_i| = m$, dla wszystkich $1 \leq i \leq \delta - 1$.

Dla dowolnych liczb całkowitych $1 \leq i < j \leq \delta - 1$ oraz dowolnej liczby naturalnej s twierdzimy też, że $j \not\equiv_{q^m-1} q^s i$. W przeciwnym razie mielibyśmy

$$j - i \equiv_{q^m-1} (q^s - 1)i$$

Jako że $q - 1 | q^m - 1$ oraz $q - 1 | q^s - 1$, dostajemy, że $q - 1 | i - j$, co jest sprzeczne z założeniem, że $\mathrm{nwd}(j - i, q^m - 1) = 1$. Czyli C_i i C_j są rozłączne. □

Lemat 12.12. *Warstwy q-cyklotomiczne modulo $q^m - 1$ dla q parzystego (czyli ciele charakterystyki 2) C_i oraz C_{2i} są takie same.*

Prosty dowód pozostawiamy jako ćwiczenie.

W takim razie w naszym oszacowaniu na wymiar możemy pominąć C_{2i}.

Lemat 12.13. *Ścisłe kody BCH długości $n = 2^m - 1$ dla zakładanej odległości 2δ oraz $2\delta + 1$ są takie same.*

Ścisły binarny kod BCH długości $n = 2^m - 1$ dla zakładanej odległości $\delta = 2t + 1$ ma wymiar co najmniej $n - m(\delta - 1)/2$.

Ponieważ warstwy cyklotomiczne C_{2i} i C_i są takie same, dostajemy pierwszy warunek, ponadto wymiar k spełnia

$$\begin{aligned} k &= q^m - 1 - \left| \bigcup_{i=1}^{2t+1} C_i \right| \\ &\geq q^m - 1 - \sum_{i=1}^{t} |C_i| \\ &\geq q^m - 1 - \sum_{i=1}^{t} m \\ &= q^m - 1 - tm \\ &= q^m - 1 - \frac{\delta - 1}{2} m \end{aligned}$$

12.6 Dekodowanie

Algorytm Berlekampa-Massey'a i algortym Forney'a stosują się do kodów BCH (bo potrzebujemy tylko postaci macierzy parzystości). Tak naprawdę to wszystkie te algorytmy były tworzone z myślą o kodach BCH, a nie o kodach RS.

12.7 Kody RS jako kody BCH

Kody RS (podklasę zdefiniowaną przez ewaluowanie w punktach będących kolejnymi potęgami generatora) można zinterpretować jako kody BCH, jeśli przyjmiemy $m = 1$.

Zauważmy, że dla $m = 1$ mamy, że wielomian minimalny elementu α^i to po prostu $X - \alpha^i$ i że są one różne dla różnych potęg α.

Twierdzenie 12.14. *q-arny kod BCH dla elementu pierwotnego $\alpha \in \mathbb{F}_q$ oraz $a = 1$ i planowanej odległości δ jest kodem RS.*

Dowód. Zauważmy, że wielomian generujący to

$$\text{nww}\left(\{X - \alpha^i\}_{i=a}^{a+\delta-2}\right) = \prod_{i=a}^{a+\delta-2} X - \alpha^i$$

Zauważmy, że „pseudo" macierz parzystości jest w tym wypadku macierzą parzystości (zgadza się w szczególności stopień). Macierz ta jest postaci

$$\begin{bmatrix} 1 & \alpha^a & \alpha^{2a} & \dots & \alpha^{(n-1)a} \\ 1 & \alpha^{a+1} & \alpha^{2(a+1)} & \dots & \alpha^{(n-1)(a+1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{a+\delta-2} & \alpha^{2(a+\delta-2)} & \dots & \alpha^{(n-1)(a+\delta-2)} \end{bmatrix}$$

Dla $a = 1$ to jest macierz parzystości kodu RS, czyli nasz kod to prawdziwy kod RS. □

Możemy wprost pokazać, że to kod MDS: stopień wielomianu generującego to $\delta - 1$, dla długości $n = q-1$ mamy, że wymiar to $n-(\delta-1)$, a odległość to przynajmniej δ (tu korzystamy z tego, że znamy ogólne oszacowanie dla kodów BCH), czyli $= \delta$, czyli są MDS. Takie kody również są nazywane kodami RS, choć to inne rozszerzenie.

Rozdział 13

Skonkatenowany kody korekcji błędów

Na podstawie [8, Rozdział 10], bardzo podobnie jest to zrobione w [9, Rozdział 6]. Trochę inaczej w [5]

Z praktycznego punktu widzenia, będziemy używali kodów pewnej ustalonej, nie za dużej długości (DVD używa około $n = 2000$). Z teoretycznego punktu widzenia jesteśmy zainteresowani również asymptotyką. Parametry, które są dla nas ważne, to zawartość informacyjna R oraz odległość; jednak jej wartość bezwzględna rośnie z n, dlatego patrzymy na $\delta(C) = \frac{\mathrm{d_H}(C)}{n}$. Większość ograniczeń da się przepisać w kategorii zależności jednego i drugiego (np. ograniczenie Singletona to $n - k + 1 \leq d \implies \delta + R \leq 1 + o(n)$.)

Popatrzmy, co pokazaliśmy:

Kod	R	δ
Hamming	$1 - \mathcal{O}(\log n/n)$	$\mathcal{O}(1/n)$
Hadamard	$\mathcal{O}(\log n/n)$	$1/2$
BCH	$\sim 1 - \frac{\delta \log n}{2}$	$\sim \delta$
Reed-Solomon	R	$> 1 - R$
Reed-Solomon*	R	$\sim (1 - R)/\log n$

Kody Hadamarda mają $\delta = \frac{1}{2}$ oraz $n \to \infty$ daje $R \to 0$. Kody Hamming $R \to 1$, ale $\delta \to 0$. Dla kodów BCH: $R \approx 1 - \frac{\log n\delta}{2}$, czyli ustalenie R również znajdziemy daje $\delta \to 0$ (i odwrotnie) Wiemy, że są dobre kody (ograniczenie dolne GV, Rozdział 4.3), ale ono nie daje jawnej konstrukcji. Ale wszystkie, które umiemy jawnie skonstruować, mają albo $\delta \to 0$ albo $R \to 0$.

13.1 Wprawka: binaryzacja kodów RS

Rozważmy kody Reeda-Solomona nad $\mathbb{F}_{2^s}$ dla dużych s. Istnieje $[n, \frac{n}{2}, \frac{n}{2}+1]$ kod Reed-Solomona (tj. $R = \frac{1}{2}$). Teraz rozważamy słowo kodowe Reed-Solomona, gdzie element $\mathbb{F}_{2^s}$ traktujemy jako wektor s bitowy, tj. element z $\mathbb{F}_2^s$. Uzyskujemy w ten sposób binarny kod liniowy o parametrach $[ns, \frac{ns}{2}, \frac{n}{2} + 1]_2$-kod. Zauważmy, że δ tego kodu to $\Theta(1/\log(ns))$, czyli bardzo mało. Dowód pozostawiamy jako proste ćwiczenie. Powodem „słabej" odległości jest to, że wektory bitów odpowiadające dwóm różnym symbolom w $\mathbb{F}_{2^s}$ mogą się różnić o jedynie jeden bit. Tak więc d różnych pozycji (z $\mathbb{F}_{2^s}$) może przekładać się na jedynie d różnic w wektorach s-bitowych. Aby to naprawić, możemy rozważyć przekształcenie tych wektorów tak aby odległość między nimi się zwiększyła. Czyli ponownie zastosować kod korekcyjny!

Ta rekurencyjna konstrukcja znana jest jako konkatenacja kodów (concatenated code) i pochodzi od Forney'a.

13.2 Konkatenacja kodów

Formalnie kod jest podzbiorem przestrzeni, ale możemy naturalnie myśleć o $[n,k]_q$-kodzie jako o funkcji z $\mathbb{F}_{q^k} \to \mathbb{F}_{q^n}$ i takie podejście jest tu bardziej przydatne.

Dla $q \geq 2$, $k \geq 1$ i $Q = q^k$, rozważmy dwa kody, które nazywamy *kodem zewnętrznym* i *wewnętrznym*;

$$C_{\text{out}} : \mathbb{F}_Q^K \to \mathbb{F}_Q^N, C_{\text{in}} : \mathbb{F}_q^k \to \mathbb{F}_q^n.$$

Wtedy konkatenacja $C_{\text{out}} \circ C_{\text{in}}$ jest zdefiniowany jako

$$\vec{m} = (m_1, \ldots, m_K) \in \mathbb{F}_Q^K \mapsto (C_{\text{in}}(C_{\text{out}}(\vec{m})_1), \ldots, C_{\text{in}}(C_{\text{out}}(\vec{m})_N)) \ .$$

To daje przekształcenie $C_{\text{out}} \circ C_{\text{in}} : \mathbb{F}_Q^K \to \mathbb{F}_q^{Nn}$ a po utożsamieniu $\mathbb{F}_q^k$ z $\mathbb{F}_Q$ (ważne, by to włożenie było $\mathbb{F}_q$-liniowe) daje

$$C_{\text{out}} \circ C_{\text{in}} : \mathbb{F}_q^{kK} \to \mathbb{F}_q^{Nn}$$

Twierdzenie 13.1. *Jeżeli C_{out} jest $[N,K,D]_{q^k}$-kodem a C_{in} jest $[n,k,d]_q$-kodem, to $C_{\text{out}} \circ C_{\text{in}}$ jest $[nN, kK, dD]_q$-kodem. W szczególności, jeżeli C_{out} (C_{in}) ma zawartość informacyjną R (r) i względna odległość δ_{out} (δ_{in}), to $C_{\text{out}} \circ C_{\text{in}}$ ma zawartość informacyjną Rr i względną odległość $\delta_{in} \cdot \delta_{out}$.*

Dowód. To przekształcenie jest $\mathbb{F}_q$-liniowe jako złożenie przekształceń liniowych.

Analogicznie pokazujemy, że jest różnowartościowe, tak więc długość i wymiar wynikają bezpośrednio z definicji.

Sprawdzenie, że odległość faktycznie wynosi przynajmniej dD pozostawiamy jako ćwiczenie. □

Uwaga. Przypomnijmy, że produkt tensorowy kodów miał takie same parametry.

13.3 Ograniczenie Zyablov'a

Teraz podstawimy konkretne kody w Twierdzeniu 13.1, co da dolne ograniczenie na zależność między sprawnością kodu a jego względną odległością; ograniczenie to znane jest jako ograniczenie Zyablov'a.

Jako kod zewnętrzny weźmy kod Reeda-Solomona. Ustalmy R, wtedy $\delta_{\text{out}} > 1 - R$, bo kody Reeda-Solomona spełniają ograniczenie Singleton'a. Teraz potrzebujemy dobrego kodu wewnętrznego, aby stworzyć dobry kod! Na szczęście kod wewnętrzny ma małą długość i możemy skonstruować go w czasie wykładniczym. Weźmy więc C_{in} jako kod spełniający ograniczenie GV o zawartości informacyjnej r.

Przypomnijmy to ograniczenie, podane w języku r, δ:
Dla $q \geq 2$, każdych $0 \leq \delta < 1 - 1/q$ oraz $0 < \epsilon \leq 1 - H_q(\delta)$, istnieje kod o zawartości informacyjnej $R \geq 1 - H_q(\delta) - \epsilon$ i odległości względnej δ.

Tak wiec jego odległość względna kodu (wewnętrznego) o zadanym r to $\delta_{\text{in}} \geq H_q^{-1}(1-r) - \epsilon$, dla pewnego $\epsilon > 0$. Mówiąć ściślej, dla każdego r, dla każdego (dostatecznie małego) $\epsilon > 0$ istnieje kod o zawartości informacyjnej $\geq r$ oraz odległości względnej przynajmniej $\delta \geq H_q^{-1}(1-r) - \epsilon$.

Podstawiając de wartości (dla kodu wewnętrznego) do Twierdzenia 13.1 otrzymujemy, że $C_{\text{out}} \circ C_{\text{in}}$ma zawartość informacyjną rR oraz odległość względną

$$\delta \geq (1 - R)(H_q^{-1}(1 - r) - \epsilon) \ .$$

Wyrażając R jako funkcję δ i r, otrzymujemy:

$$R \geq 1 - \frac{\delta}{H_q^{-1}(1-r) - \epsilon} \ .$$

Możemy teraz zoptymalizować po r (nie jest ustalone w żaden sposób).

$$\mathcal{R} \geq \max_{0<r<1-H_q(\delta+\epsilon)} r\left(1 - \frac{\delta}{H_q^{-1}(1-r) - \epsilon}\right)$$

gdzie ograniczenie $r < 1 - H_q(\delta + \epsilon)$ jest konieczne, by zapewnić, że $R > 0$. Ta ograniczenie dolne znane jest jako ograniczenie (dolne) Zyablov'a.

Twierdzenie 13.2 (Ograniczenie (dolne) Zyablov'a)**.** *Dla każdego δ oraz każdego $\epsilon > 0$ istnieje kod q-arny o zawartości informacyjnej przynajmniej*

$$\mathcal{R} \geq \max_{0<r<1-H_q(\delta+\epsilon)} r\left(1 - \frac{\delta}{H_q^{-1}(1-r) - \epsilon}\right)$$

Sprawdźmy np. $\delta = \frac{1}{2} - \epsilon$. Wychodzi wtedy $R \geq \Omega(\epsilon^3)$. (Łatwy rachunek pokazuje, że ograniczenie GV daje $\Omega(\epsilon^2)$.) Dowód pozostawiamy jako ćwiczenie.

W szczególności ograniczenie Zyablov'a implikuje, że dla każdego $\delta > 0$ istnieje skonkatenowany kod o zawartości informacyjnej $R > 0$. Ale w sumie to już wiedzieliśmy bezpośrednio z ograczenia GV.

Naturalne jest pytanie, czy możemy taki kod (najlepiej liniowy) podać podać „wprost" (i co to w zasadzie znaczy dokładnie).

13.4 Jawnie zadane kody

Chcemy, aby kod był podany jawnie, w najsłabszym sensie: potrafimy go policzyć w czasie wielomianowym.

Niech C będzie $[N, K]_Q$ kodem Reeda-Solomona, gdzie $N = Q-1$ (czyli obliczamy wielomian w $\mathbb{F}_Q$, gdzie $Q = q^k$). Oznacza to, że $k = \Theta(\log_q N)$. Potrzebujemy wydajnej konstrukcji kodu wewnętrznego, który spełnia z równością ograniczenie GV. Nie da się raczej tego zrobić w czasie wielomianowym od k, bo to ogólnie wyeliminowało cały problem z niekonstrukcyjnością ograniczenia GV, ale skoro $k = \mathcal{O}(\log_q N)$, to wystarczy nam czas wykładniczy od k, co wciąż jest wielomianowe od N.

Możliwe są dwa podejścia do konstrukcji C_{in}w czasie wykładniczym:

- Przejrzeć wszystkie macierze generujące spełniające warunki (tj. odpowiedniego rozmiaru i nad właściwym ciałem). Ograniczenie GV pokazuje, że istnieje taki kod liniowy. To zajmie $q^{\mathcal{O}(kn)}$ czasu. Podstawiając $k = rn$ (i wiedząc, że celujemy w stałe r), otrzymujemy
$$q^{\mathcal{O}(kn)} = q^{\mathcal{O}(k^2)} = N^{\mathcal{O}(\log_q N)}$$
co jest czasem quasi-wielomianowym.

- Można też skonstruować C_{in}w czasie $\mathcal{O}(poly(n)q^n) = \mathcal{O}(poly(n)q^{rn}) = \mathcal{O}(poly(n)N^r)$: w dowodzie ograniczenie GV (w wersji Verschamova) konstruowaliśmy macierz parzystości tak, że dowolne d jej kolumn jest liniowo niezależnych. Dodając kolejną kolumnę musimy przejrzeć $q^{n-k} = q^{(1/r-1)k} \approx N^{1/r-1}$ kandydatów. Dla kandydata musimy przejrzeć, czy jest liniowo niezależny z dowolnymi spośród poprzednich $d-1$ kolumn, takich podzbiorów jest mniej niż $2^n < q^n = q^{1/rk} \approx N^{1/r}$. Dla każdego z nich w czasie wielomianowym od n spradzamy, czy są liniowo niezależne.

Twierdzenie 13.3. *Możemy skonstruować w czasie wielomianowym kod, który spełnia ograniczenie Zyablov'a.*

13.5 Kod silnie jawny: kod Justesena

Przykrym aspektem tej konstrukcji jest przeszukanie wszystkich możliwych kodów wewnętrznych. Naturalne jest pytanie, czy da sie to zrobić jeszcze lepiej? Odpowiedź jest twierdząca: jest nią *kod Justesena*. Ściśle rzecz ujmując, kod jest silnie jawny, jeśli potrafimy nim zakodować używając $polylog(n)$ operacji na jeden element wejścia (albo potrafimy„mały" układ arytmetyczny kodujący zrobić).

Głównym pomysł jest taki, że skoro chodzi nam tylko o to, by kod był asymptotycznie dobry, to poprzedni argument (kod RS + kod spełniający ograniczenie GV) działa też wtedy, gdy

1. wybierzemy różne kody wewnętrzne dla każdego z N zewnętrznych słów kodowych.
2. *większość* (ale niekoniecznie wszystkie) z użytych kodów wewnętrznych spełnia ograniczenie GV z równością.

Okazuje się, że konstruowanie zbioru kodów w taki sposób, że większość z nich spełnia ograniczenie GV z równością jest dużo łatwiejsza, niż skonstruowanie pojedynczego kodu spełniającego tę równość (co jest jakoś intuicyjnie zrozumiałe: losowo wybrany kod ją spełnia, trzeba tylko wybrać dostatecznie wiele kodów). W szczególności zbiór wszystkich kodów liniowych spełnia ten warunek (to jest dokładnie dowód ograniczenia GV), ale tak zdefiniowana rodzina jest za duża. Kod Justesena jest połączony z wieloma, różnymi kodami wewnętrznymi. Konkretnie, składa się z kodu zewnętrznego C_{out} będącego $[N, K, D]_{q^k}$-kodem oraz *różnych* kodów wewnętrznych ${C_{\text{in}}}^i$ dla $1 \leq i \leq N$. Formalnie, konkatenacja tych kodów, oznaczona przez $C_{\text{out}} \circ ({C_{\text{in}}}^1, \ldots, {C_{\text{in}}}^N)$ jest zdefiniowana w następujący sposób: dla wiadomości $\vec{m} \in \mathbb{F}_{q^k}^K$ niech $(c_1, \ldots, c_N) = C_{\text{out}}(\vec{m})$ będzie wynikiem zastosowania kodu zewnętrznego. Wtedy

$$C_{\text{out}} \circ ({C_{\text{in}}}^1, \ldots, {C_{\text{in}}}^N)(\vec{m}) = ({C_{\text{in}}}^1(c_1), {C_{\text{in}}}^2(c_2), \ldots, {C_{\text{in}}}^N(c_N))$$

Potrzebujemy udowodnić następujące twierdzenie:

Twierdzenie 13.4. *Niech $\epsilon > 0$. Istnieje rodzina kodów wewnętrznych ${C_{\text{in}}}^1, \ldots, {C_{\text{in}}}^N$, każdy o zawartości informacyjnej $\frac{1}{2}$, gdzie $N = q^k - 1$, taka że dla co najmniej $(1-\epsilon)N$ wartości i kod ${C_{\text{in}}}^i$ ma względną odległość $\delta({C_{\text{in}}}^i) \geq H_q^{-1}(\frac{1}{2} - \epsilon)$.*

Rodzina ta wygląda w następująco: dla $\alpha \in \mathbb{F}_{q^k}^*$, kod wewnętrzny ${C_{\text{in}}}^\alpha : \mathbb{F}_q^k \to \mathbb{F}_q^{2k}$ interpretujemy jako funkcję ${C_{\text{in}}}^\alpha : \mathbb{F}_{q^k} \to \mathbb{F}_{q^k}^2$ i definiujemy ją jako

$${C_{\text{in}}}^\alpha(x) = (x, \alpha x)$$

Rodzinę taką nazywamy *rodziną Wozencrafta*. Każde ${C_{\text{in}}}^\alpha$ jest liniowe i jest łatwe do wyliczenia.

Zaś sama rodzina to $\{{C_{\text{in}}}^\alpha\}_{\alpha \in \mathbb{F}_{q^k}^*}$, czyli ma $q^k - 1 = N$ elementów, tak jak trzeba.

Dowód. Ustalmy $\vec{y} = (y_1, y_2) \in \mathbb{F}_{q^k}^2 \setminus \{0, 0\}$. Zauważmy, że oznacza to, że $y_1 = 0 = y_2$ nie jest możliwe. Chcemy pokazać, że $\vec{y} \in {C_{\text{in}}}^\alpha$ dla najwyżej jednego $\alpha \in \mathbb{F}_{q^k}^*$; pozostawimy to jako ćwiczenie.

Rozważmy $\vec{y}$ takie że $\|\vec{y}\|_1 < H_q^{-1}\left(\frac{1}{2} - \epsilon\right) n$. Zauważmy, że jeśli $\vec{y} \in {C_{\text{in}}}^\alpha$ to ${C_{\text{in}}}^\alpha$ jest zły, bo ma względną odległość mniejszą niż $H_q^{-1}\left(\frac{1}{2} - \epsilon\right) n$. Ale z pokazanej przed chwilą własności, jeden taki (zły) $\vec{y}$ należy do najwyżej jednego kodu wewnętrznego (tzn. dla najwyżej jednej α Tak wiec sumaryczna liczba złych kodów to najwyżej

$$\{\vec{y} : \|\vec{y}\|_1 < H_q^{-1}\left(\frac{1}{2} - \epsilon\right) 2k\} \leq \text{Vol}\left(H_q^{-1}\left(\frac{1}{2} - \epsilon\right) 2k, 2k\right)$$

Przypomnijmy oszacowanie na objętość kuli:

$$\mathrm{Vol}_q(pn, n) \leq q^{H_q(p)n}$$

Co daje

$$\begin{aligned}
\mathrm{Vol}\left(H_q^{-1}\left(\frac{1}{2}-\epsilon\right)2k, 2k\right) &\leq q^{H_q(H_q^{-1}(\frac{1}{2}-\epsilon))\cdot 2k} \\
&= q^{\left(\frac{1}{2}-\epsilon\right)\cdot 2k} \\
&= \left(q^k\right)^{1-2\epsilon} \\
&= (1+N)^{1-2\epsilon} \\
&< \epsilon N \qquad \text{Dla dostateczni dużych } N.
\end{aligned}$$

Zatem dla co najmniej $(1-\epsilon)N$ wartości α, kod ${C_{\mathrm{in}}}^{\alpha}$ ma względną odległość co najmniej $H_q^{-1}(\frac{1}{2}-\epsilon)$. □

Kod Justesena jest zdefiniowany następująco: kod zewnętrzny C_{out} to kod Reeda-Solomona z punktami ewaluacji $\mathbb{F}_{q^k}^*$ i zawartości informacyjnej R, $0 < R < 1$. Kod zewnętrzny C_{out} ma względną odległość $\delta_{\mathrm{out}} = 1 - R$ i długość bloku $N = q^k - 1$. Rodzina kodów wewnętrznych to rodzina Wozencrafta $\{{C_{\mathrm{in}}}^{\alpha}\}\alpha \in \mathbb{F}_{q^k}^*$. Kod Justesena to

$$C^* = C_{\mathrm{out}} \circ ({C_{\mathrm{in}}}^1, \ldots, {C_{\mathrm{in}}}^N) \ .$$

Z definicji kod Justesena C^* ma zawartość informacyjną $\frac{R}{2}$. Oszacujmy jego odległość względną.

Twierdzenie 13.5. *Niech $\epsilon > 0$. Kod Justesena C^* ma względną odległość nie mniejszą niż*

$$(1 - R - \epsilon) \cdot H_q^{-1}\left(\frac{1}{2} - \epsilon\right) \ .$$

Dowód. Rozważmy $\vec{m} \neq \vec{m'} \in \mathbb{F}_{q^k}^K$. Niech S: zbiór wartości kodu zewnętrznego różnych dla $\vec{m}$ i $\vec{m'}$, tj.

$$S = \{i \ : \ C_{\mathrm{out}}(\vec{m})_i \neq \ C_{\mathrm{out}}(\vec{m'})_i\} \ .$$

Jako że odległość względna kodu zewnętrznego to przynajmniej $(1 - R)$, dostajemy

$$|S| \geq (1 - R)N \ ,$$

Nazwiemy i dobrym, jeśli $\|{C_{\mathrm{in}}}^i\|_1 \geq \ H_q^{-1}(\frac{1}{2} - \epsilon) \cdot 2k = d$, w przeciwnym przypadku i jest złe. Z Twierdzenia 13.4 istnieją najwyżej ϵN złych i. Niech S_g to i takie że $i \in S$ oraz ${C_{\mathrm{in}}}^i$ jest dobrym kodem, zaś S_b: złym. Czyli $S_b \leq \epsilon N$. I w takim razie

$$|S_g| = |S| - |S_b| \geq (1 - R - \epsilon)N \ .$$

Dla każdego $i \in S_g$ mamy z definicji

$$\mathrm{d_H}\left({C_{\mathrm{in}}}^i(C_{\mathrm{out}}(\vec{m})_i), {C_{\mathrm{in}}}^i(C_{\mathrm{out}}(\vec{m'})_i)\right) \geq d \ .$$

Czyli odległość $\mathrm{d_H}(C^*)$ wynosi co najmniej

$$|S_g|d \geq= (1 - R - \epsilon)NH_q^{-1}\left(\frac{1}{2} - \epsilon\right) \cdot 2k \ ,$$

po podzieleniu przez $2Nk$, czyli długość kodu, dostajemy

$$\delta \geq (1 - R - \epsilon)H_q^{-1}\left(\frac{1}{2} - \epsilon\right) \ ,$$

czego należało dowieść. □

Ponieważ kody Reed-Solomona, jak również rodzina Wozencrafta są łatwe do obliczenia, to jest taki również kod Justesena.

Rozdział 14

(Efektywne) dekodowanie kodów skonkatenowanych

Chcielibyśmy (efektywnie) dekodować kod skonkatenowany, dla ustalenia parametrów, powiedzmy dla $C_{\text{out}} \circ C_{\text{in}}$składającego się z kodu zewnętrznego C_{out}będącego $[N, K, D]_Q$ kodem i kodu wewnętrzengo C_{in}, będącego $[n, k, d]_q$-kodem, gdzie $Q = q^k$; ponadto $Q = \mathcal{O}(N)$.

14.1 Algorytm naturalny (naiwny)

Naturalny algorytm dekodujący jest następujący: najpierw dekodujemy kod wewnętrzny, dla każego słowa z osobna, a następnie kod zewnętrzny.

Załóżmy na razie, że mamy algorytm dekodowanie (do teoretycznej granicy $D/2$) dla kodu zewnętrznego, jest to prawda dla zastosowanego przez nas kodu RS. To pozostawia nam zadanie wymyślenia wielomianowego algorytmu dekodowania dla kodu wewnętrznego (zwłaszcza w przypadku kodu Justesena, który używa *różnych* kodów wewnętrznych). Na szczęście, czas działania ma być wielomianowy od ostatecznej długości bloku, która jest *wykładnicza* od długości bloku kodu wewnętrznego. Tak więc przejrzenie *wszystkich* słów kodowych i wybranie najbliższego jest OK (czas działania: $N \cdot q^k \approx N^2$). Czyli całkowity czas działania jest wielomianowy (bo kod zewnętrzny też dekodujemy w czasie wielomianowym).

Zobaczmy teraz, ile błędów potrafi poprawić ten kod:

Twierdzenie 14.1. *Naturalny algorytm korekcji błędów dla kodów skonkatenowanych poprawia $< Dd/4$ błędów.*

Dowód. Załóżmy, że otrzymaliśmy wiadomość (z błędami) $\vec{y}$, niech $\vec{c}$ będzie słowem kodowym odpowiadającym oryginalnej wiadomości, w szczególności $\mathrm{d_H}(\vec{y}, \vec{c}) < Dd/4$.

Oznaczmy $\vec{x}$: otrzymana wiadomość ($\vec{y}$) po pierwszym etapie dekodowania oraz x': $\vec{c}$ po pierwszym etapie dekodowania (czyli oryginalna wiadomość po pierwszym etapie kodowania). Skoro źle zdekodujemy całość, to $\mathrm{d_H}(\vec{x}, \vec{x'}) \geq D/2$ (być możę nawet więcej, ale implikacja jest prawdziwa). Ale każda błędna pozycja oznacza, że na pierwszym etapie było tam przynajmniej $d/2$ błędów. Czyli musiało być przynajmniej $dD/4$ błędów. Przez kontrapozycję dostajemy tezę. □

14.2 Ogólne dekodowanie

Wadą algorytmu naturalnego jest to, że nie uwzględnia on informacje dostarczanych przez pierwszy etap dekodowania: na przykład, nie rozróżnia się w niej sytuacji, w których kod wewnętrzny otrzymanego słowa ma odległość Hamming jeden (od słowa kodowego) od przypadku,

gdy ma odległość (prawie) połowę odległości kodu. Naturalnym wydaje się korzystanie z tych informacji.

Podamy algorytm znany jako *uogólniony algorytm najmniejszej odległości* (GMD), który wykorzystuje te dodatkowe informacje. Od teraz zakładamy, że kod zewnętrzny C_{out}jest $[N, K, D]_q^k$ kodem, który można zdekodować w czasie wielomianowym, również dla błędów wymazań, do granicy

$$2e + s < d$$

To ograniczenie jest prawdą dla kodów RS:

Twierdzenie 14.2. $[N, K, N-K+1]$ *kod Reeda-Solomona może być poprawiony w czasie* $\mathcal{O}(N^3)$ *dla* e *błędów i* s *błędów wymazań, jeśli*

$$2e + s < N - K + 1 \ .$$

Prosty dowód pozostawiamy jako ćwiczenie.

Zanim podamy algorytm, przyjrzyjmy się dwóm szczególnym przypadkom, aby wyrobić sobie właściwą intuicję: Rozważmy otrzymane słowo $\vec{y} = (y_1, \ldots, y_N) \in \mathbb{F}_{q^n}^N$ oraz najbliższe słowo kodowe $\vec{c} = (c_1, \ldots, c_N)$ o następującej właściwości: dla każdego $1 \leq i \leq N$, albo $y_i = c_i$ albo $\mathrm{d_H}(y_i, c_i) \geq d/2$. Jeśli $\mathrm{d_H}(\vec{y}, \vec{c}) < dD/2$ i wymażemy wszystkie pozycje nie będące słowami kodowymi, gdzie pojawił się błąd, to będziemy w stanie odtworzyć poprawne słowo kodowe. Każda wymazana pozycja to przynajmniej $d/2$ błędów, każda błędna niewymazana to przynajmniej d błędów. Jeśli takich pozycji jest e, s, to jako że w sumie błędów jest najwyżej $dD/2$, dostajemy

$$e \cdot d + s \cdot \frac{d}{2} < dD/2$$

błędów. Po podzieleniu przez $d/2$ dostajemy

$$2e + s < D \ ,$$

czyli potrafimy to poprawnie zdekodować.

Rozważmy teraz otrzymane słowo $\vec{y} = (y_1, \ldots, y_N)$ o tej własności, że jeśli w którymś ze słów kodowych kodu wewnętrznego wystąpił błąd, to i tak dostaliśmy poprawne słowo kodowe, czyli jest tam przynajmniej d błędów. Czyli jeśli nic nie poprawimy, to będzie mniej niż $D/2$ błędów dla kodu zewnętrznego, który poprawnie wszystko zdekoduje.

Nasz algorytm będzie uogólniał obie te obserwacje, tak by płynnie interpolować te sytuacje. W szczególności, dla skrajnych przypadków będzie działa tak, jak podaliśmy powyżej.

Możemy spojrzeć na problem jeszcze inaczej: powiedzmy, że elementom kodu zewnętrznego po zdekodowaniu (i wymazaniu) słów kodu wewnętrznego przypiszemy wagi: 0 jeśli jest to właściwy element (czyli bez błędu), $\frac{d}{2}$ jeśli jest ono wymazane oraz d, jeśli jest błędnie zdekodowane przez kod wewnętrzny. Łatwo zauważyć, że poprawnie zdekodujemy kod zewnętrzny, jeśli tak zdefiniowana waga jest mniejsza niż $Dd/2$. Waga ta w pewnym sensie odpowiada liczbie błędów (w sumie) w kodzie wewnętrznym, dla niego też możemy myśleć, że słowa mają wagi = liczba błędów w nich. Przy czym:

- Jeśli wymażemy słowo, to zmieniamy jego wagę na $d/2$, niezależnie od tego, ile wynosiła; czyli polepszamy dla słów mających $\geq d/2$ błędów i *pogarszamy* dla tych z $< d/2$.
- Jeśli poprawimy, to dla słów z liczbą błędów $< d/2$ zmniejszamy liczbę błędów, zaś dla tych z $\geq d/2$

Te, które mają $\geq d/2$ należy wymazać, a te z $< d/2$ należy poprawić. Tylko nie wiadomo, które jest które... Co więcej, ustalenie arbitralnej wartości nie może nam dać dobrego wyniku, bo zawsze możemy dać coś tuż poniżej lub powyżej progu (ćwiczenie).

Ale może możemy wylosować?

14.3 GMD: pierwsza wersja

Algorytm dekodowania kodu wewnętrznego będzie następujący:

Algorytm 5 Algorytm GMD (wersja I)

Założenie: Otrzymane słowo to $\vec{y} = y_1, \ldots, y_N \in \mathbb{F}_{q^n}^N$
  for $i \leftarrow 1 \,..\, N$ **do**
    $y'_i \leftarrow$ słowo kodowe najbliższe do y_i
    $w_i \leftarrow \min(\mathrm{d_H}(y'_i, y_i), d/2)$
    z prawdopodobieństwem $\frac{2w_i}{d}$ ustaw $x'_i \leftarrow ?$
    w pozostałym przypadku ustaw x_i jako odkodowane y'_i
  return zdekodowane $\vec{x}$

Pokażemy, najpierw, że wartość oczekiwana suma błędów i (podwójnej liczby) wymazań, jest $\leq D$, czyli jesteśmy w stanie zdekodować.

Lemat 14.3. *Niech $\vec{y}$ będzie otrzymanym słowem, przy czym istnieje (jedyne) słowo kodowe $C_{\mathrm{out}} \circ C_{\mathrm{in}}(\vec{m}) = (c_1, \ldots, c_N) \in \mathbb{F}_{q^n}^N$ takie, że $\mathrm{d_H}(C_{\mathrm{out}} \circ C_{\mathrm{in}}(\vec{m}), \vec{y}) < Dd/2$, niech e, s oznaczają liczbę błędów oraz wymazań w słowie $\vec{x}$ (w porównaniu z $C_{\mathrm{out}}(\vec{m})$). Wtedy*

$$\mathbb{E}[2e + s] \leq D$$

Zauważ, że jeśli $2e + s < D$, to zgodnie z Twierdzeniem 14.2, algorytm poprawnie odtworzy oryginalną wiadomość.

Dowód. Dla każdego $1 \leq i \leq N$ definiujemy $e_i = \mathrm{d_H}(y_i, c_i)$. Zauważmy, że

$$\sum_{i=1}^{N} e_i < \frac{Dd}{2}$$

Dla każdego $1 \leq i \leq N$ definiujemy dwie 0/1 zmienne losowe:

$$X_i^? = \begin{cases} 1 & \text{jeśli } x_i = ? \\ 0 & \text{w przeciwnym przypadku} \end{cases} \qquad X_i^e = \begin{cases} 1 & \text{jeśli } x_i \neq ? \text{ i zostało źle zdekodowane} \\ 0 & \text{w przeciwnym przypadku} \end{cases}$$

Wystarczy w takim razie, że pokażemy, że dla każdego $1 \leq i \leq N$ zachodzi

$$\mathbb{E}[2X_i^e + X_i^?] \leq \frac{2e_i}{d} \tag{14.1}$$

Faktycznie: z definicji mamy: $e = \sum_{i=1}^{N} \mathbb{E}[X_i^e]$ oraz $s = \sum_{i=1}^{N} \mathbb{E}[X_i^?]$. I wtedy z liniowości wartości oczekiwanej dostajemy

$$\mathbb{E}[2e + s] \leq \frac{2}{d} \sum_{i=1}^{n} e_i < D$$

czyli średnią wartość potrafimy zdekodować.

Pokażemy nierówność (14.1) przez rozpatrzenie przypadków: Ustalmy dowolne $1 \leq i \leq N$.

mało błędów Rozpatrzmy przypadek, że zdekodujemy y_i (do x_i), to otrzymamy poprawne słowo kodowe, tzn. wartość kodu zewnętrznego na m_i. Zauważmy, że liczba błędów może wciąż być w tym wypadku większa, niż $d/2$.

Zauważmy najpierw, że skoro dobrze zdekodujemy, to $X_i^e = 0$. Wtedy

$$\mathbb{E}[X_i^?] = \mathbb{P}[X_i^? = 1] = \frac{2w_i}{d} \quad \text{oraz} \quad \mathbb{E}[X_i^e] = \mathbb{P}[X_i^e = 1] = 0 \ .$$

Ponadto z definicji mamy

$$w_i = \min\left(\mathrm{d_H}(y_i, y_i'), \frac{d}{2}\right) \leq \mathrm{d_H}(y_i, c_i) = e_i$$

Czyli

$$\mathbb{E}[2X_i^e + X_i^?] = \frac{2w_i}{d} \leq \frac{2e_i}{d} \ ,$$

co należało pokazać.

dużo błędów Rozpatrzmy teraz przypadek, że x_i, które zdekodujemy, nie jest zakodowanym (przez C_{out}) częścią wiadomości m_i. W szczególności, jest $\geq \frac{d}{2}$ błędów. Tak jak w poprzednim przypadku, nadal mamy

$$\mathbb{E}[X_i^?] = \frac{2w_i}{d}$$

Ale jeśli nie wymazaliśmy pozycje, to mamy $X_i^e = 1$. Tak więc

$$\mathbb{E}[X_i^e] = \mathbb{P}[X_i^e = 1] = 1 - \frac{2w_i}{d} \ .$$

Pokażemy za chwilę, że

$$e_i + w_i \geq d \tag{14.2}$$

Jeśli to prawda, to

$$\mathbb{E}[2X_i^e + X_i^?] = 2 - \frac{2w_i}{d} \leq \frac{2e_i}{d} \ ,$$

co było do pokazania.

Aby zakończyć dowód, pozostało nam udowodnić (14.2), które ponownie robimy przez przypadki:

$w_i = \mathrm{d_H}(y_i, y_i') < d/2$ Zgodnie z definicją e_i , mamy

$$e_i + w_i = \mathrm{d_H}(y_i, c_i) + \mathrm{d_H}(y_i', y_i) \geq \mathrm{d_H}(c_i, y_i') \geq d,$$

gdzie pierwsza nierówność wynika z nierówności trójkąta, a druga z tego, ile wynosi odległość kodu (wewnętrznego).

$w_i = \frac{d}{2} \leq \mathrm{d_H}(y_i', y_i)$ Ponieważ y_i' jest najbliższym słowem kodowym, mamy

$$\frac{d}{2} < \ \mathrm{d_H}(y_i', y_i) \leq \mathrm{d_H}(c_i, y_i) = e_i$$

i w takim razie

$$e_i + w_i > d \ . \quad \square$$

Algorytm 6 Algorytm GMD (wersja II)

Założenie: Otrzymane słowo to $\vec{y} = y_1, \ldots, y_N \in \mathbb{F}_{q^n}^N$
wylosuj z rozkładem jednostajnym $\theta \in [0.1]$
for $i \leftarrow 1 \,..\, N$ **do**
$y'_i \leftarrow$ słowo kodowe najbliższe do y_i
$w_i \leftarrow \min(\mathrm{d_H}(y'_i, y_i), d/2)$
if $\frac{2w_i}{d} > \theta$ **then**
ustaw $x'_i \leftarrow ?$
else
ustaw x_i jako odkodowane y'_i
return zdekodowane $\vec{x}$

14.4 Druga wersja algorytmu

Zauważmy, że losowanie za każdym razem nie jest w zasadzie do niczego potrzebne: zamiast losować z jakimś rozkładem dla i (zależnym od w_i), możemy wylosować parametr $\theta \in [0, 1]$ i porównywać wagę z tym parametrem. Ponieważ wszystkie szacowania dotyczyły wartości oczekiwanej, to wszystko dalej jest prawdą (z liniowości wartości oczekiwanej).

Zauważmy, że losowości używaliśmy tylko wtedy, gdy liczyliśmy, że

$$\mathbb{P}[x'_i \leftarrow ?] = \frac{2w_i}{d}$$

A teraz mamy

$$\mathbb{P}[x'_i \leftarrow ?] = \mathbb{P}\left[\theta \in \left[0, \frac{2w_i}{d}\right)\right] = \frac{2w_i}{d} \ .$$

14.5 Trzecia wersja: derandomizacja

Zauważmy, że skoro w_i ma wartość dyskretną, to zamiast losować θ jednostajnie z $[0, 1]$, możemy wylosować ze skończonego zbioru $\{0, \frac{2}{d}, \frac{4}{d}, \ldots, 1\}$: każde $\theta \in \left[\frac{2i}{d}, \frac{2(i+1)}{d}\right)$ daje taki sam wynik (mała uwaga: ostatni przedział może mieć długość $1/d$ a nie $2/d$, w zależności od parzystości d). Ale skoro tak, to każdą z tych wartości możemy po prostu osobno sprawdzić.

Uwaga. To losowanie niekoniecznie powinno być jednostajne: zależy od długości przedziału, a ostatni ma różną długość, w zależności od parzystości d, ale to nie ma dla nas znaczenia, skoro i tak sprawdzamy wszystkie możliwości.

Zauważmy, że nowy algorytm to poprzedni $\approx d/2$ razy. Czyli działa on w czasie wielomianowym.

Algorytm 7 Algorytm GMD (wersja II)

Założenie: Otrzymane słowo to $\vec{y} = y_1, \ldots, y_N \in \mathbb{F}_{q^n}^N$

$S \leftarrow \emptyset$
for $\theta \in \{0, 2/d, 4/d, \ldots, 1\}$ **do**
  for $i \leftarrow 1 \,..\, N$ **do**
    $y'_i \leftarrow$ słowo kodowe najbliższe do y_i
    $w_i \leftarrow \min(\mathrm{d_H}(y'_i, y_i), d/2)$
    if $\frac{2w_i}{d} > \theta$ **then**
      ustaw $x'_i \leftarrow ?$
    else
      ustaw x_i jako odkodowane y'_i
  $S \leftarrow S \cup$ zdekodowane $\vec{x}$
return element S, dla którego po zakodowanie odległość od $\vec{y}$ jest najmniejsza

Rozdział 15

Dekodowanie do list (List decoding)

Na podstawie [8, Rozdział 7].

Motywacja:

- czasami najbliższe słowo kodowe jest dalej niż $< d/2$, a nasze algorytmy i tak nie potrafią nic zrobić. Tzn. nie potrafią nic zrobić.
- czasem mamy dodatkową informację (zarówno nieformalny kontekst jak i być może jakąś ściśle zdefiniowaną podpowiedź)

Zajmiemy się teraz dekodowaniem list (list decoding), tj. dekoder może zwrócić odpowiedź w postaci listy słów kodowych spełniających warunek. Jest to odpowiedź na powyższe problemy. Idea jest dość stara, ale algorytmy raczej świeże; choć granice dolne znane były od dawna.

Definicja 15.1 (Dekodowanie do listy)**.** Dla $0 \leq \rho \leq 1$ oraz $L \geq 1$, kod $C \subseteq \Sigma^n$ jest (ρ, L)-dekodowalny do listy, jeżeli dla każdego słowo $\vec{y} \in \Sigma^n$ zbiór słów kodowych w odległości ρn jest rozmiaru najwyżej L:

$$|\{\vec{c} : \vec{c} \in C \wedge \mathrm{d_H}(\vec{c}, \vec{y}) \leq \rho n\}| \leq L \ .$$

Dla danych: parametru błędu ρ, kodu C i odebranego słowa $\vec{y}$, algorytm dekodowania do listy powinien zwrócić wszystkie słowa kodowe w C, które znajdują się we (względnej) odległości Hamminga ρ od $\vec{y}$.

Uwaga. Ponieważ możliwe, że algorytm zwróci L słów, z algorytmicznego punktu widzenia interesuje nas sytuacja, kiedy L jest (najwyżej) wielomianowe od n (czyli długości słowa kodowego). (w przeciwnym razie algorytm będzie musiał wypisać ponad-wielomianową liczbę słów kodowych i Innym powodem, dla którego L należy związać z n, jest to, że w przeciwnym przypadku problem dekodowania do listy może się strywializować: na przykład, można wypisać wszystkie słowa kodowe w kodzie. Czas wykładniczy można też trywialnie osiągnąć: wystarczy przejrzeć wszystkie słowa kodowe.

Jeśli myślimy, że kodów używamy faktycznie do komunikacji, to dekodowanie do listy jest problematyczne. Możemy albo uznać, że jeśli jest więcej niż 1 element, to nie zwracamy nic. To uogólnia tradycyjne poprawianie błędów i (częściowo) naprawia problem nieumiejętności dekodowania najbliższego słowa, nawet jeśli jest wyznaczone jednoznacznie.

Alternatywnie, możemy myśleć, że dekoder ma dostęp do dodatkowej informacji, która pozwoli mu wybrać to jedyne słowo. Skoro elementów jest $\leq L$, to myślimy, że $\log L$ bitów starczy, co jest małe w stosunku do wielkości komunikatu. (To podejście z rejonów złożoności obliczeniowej.)

15.1 Ograniczenie Johnsona

Twierdzenie 15.2 (Ograniczenie Johnson'a). *Niech $C \subseteq \mathbb{F}_q^n$ będzie kodem, $\delta = \delta(C) = \mathrm{d}(C)/n$. Jeśli $\rho < J_q(\delta)$, to C jest $(\rho, q\delta n^2)$-dekodowalny do listy, gdzie $J_q(\delta)$ to:*

$$J_q(\delta) = \frac{q-1}{q}\left(1 - \sqrt{1 - \frac{q}{q-1}\delta}\right) \ .$$

To twierdzenie pokazuje, że dekodowanie do lit, tak jak tradycyjne poprawianie błędów, ma sens: dla dostatecznie małej wartości zawsze potrafimy to zrobić.

Dowód. Dowód pokażemy dla $q = 2$. Zauważmy, że

$$J_2(\delta) = \frac{1}{2}\left(1 - \sqrt{1 - 2\delta}\right)$$

Dowód będzie przez podwójne zliczanie.

Chcemy pokazać, że dla każdego kodu binarnego $C \subseteq \{0,1\}^n$ o odległości d i dowolnego wektora $\vec{y} \in \{0,1\}^n$ oraz $\rho < J_2(\delta)$ zachodzi

$$|B(y, \rho n) \cap C| \leq 2dn$$

($2dn$ to $q\delta n^2$ z twierdzenia po podstawieniu $q = 2$. Dla prostoty zdefiniujmy $e = \rho n$.)

Ustalmy dowolne C i $\vec{y}$. Niech $\vec{c}_1, \ldots, \vec{c}_M \in B(y, e)$. Zdefiniujmy

$$\vec{c'}_i = \vec{c}_i - \vec{y} \text{ dla } 1 \leq i \leq M \ ,$$

Zauważmy, że zbiór $\{\vec{c'}_i : \vec{c}_i \in C\}$ też jest kodem, o takich samych parametrach, jak C (jeśli C jest liniowy, to C' nie, o ile $\vec{y} \notin C$).

Od tego miejsca dowód używa podobnego podejścia, jak przy Ograniczeniu Plotkina. Tylko na koniec doliczamy wszystkie wartości dużo dalej. Pozostała część dowodu nie została pokazana na wykładzie.

Wtedy

$$\|\vec{c'}_i\|_1 \leq e \qquad \text{bo } \vec{c}_i \in B(y, e) \tag{15.1}$$

$$\mathrm{d_H}(\vec{c'}_i, \vec{c'}_j) \geq d \qquad \text{dla } i \neq j \text{ bo } \mathrm{d_H}(\vec{c}_i, \vec{c}_j) \geq d \ . \tag{15.2}$$

Zdefiniujmy

$$S = \sum_{i<j} \mathrm{d_H}(\vec{c'}_i, \vec{c'}_j) \ .$$

Pokażemy zarówno górne jak i dolne ograniczenie na S, ich porównanie da pożądane oszacowanie na M.

Z (15.2) mamy

$$S \geq \binom{M}{2} d \tag{15.3}$$

Rozważmy macierz $n \times M$ postaci

$$(\vec{c'_1} | \cdots | \vec{c'_M}) \ .$$

Niech m_i to liczba jedynek w i-tym rzędzie. Następnie i -ty rząd macierzy wnosi do S wartość $m_i(M - m_i)$, ponieważ jest to jest liczbą par $0-1$ w tym rzędzie. To oznacza, że

$$S = \sum_{i=1}^{n} m_i(M - m_i) \ . \tag{15.4}$$

Zdefiniujmy

$$\overline{e} = \sum_{i=1}^{n} \frac{m_i}{M}$$

Zauważmy, że

$$\begin{aligned} \sum_{i=1}^{n} m_i &= \sum_{i=1}^{M} \|\vec{c}_i\|_1 \\ &\leq eM \qquad \text{z (15.1)} \ . \end{aligned}$$

I w takim razie mamy

$$\overline{e} \leq e$$

Z (15.4) mamy:

$$\begin{aligned} S &= \sum_{i=1}^{n} m_i(M - m_i) \qquad (15.5) \\ &= M^2\overline{e} - \sum_{i=1}^{n} m_i^2 \end{aligned}$$

Używając swojej ulubionej nierówności dostajemy

$$\left(\frac{\sum_{i=1}^{n} m_i}{n}\right)^2 \leq \frac{\left(\sum_{i=1}^{n} m_i\right)^2}{n}$$

I dlatego

$$\begin{aligned} M^2\overline{e} - \sum_{i=1}^{n} m_i^2 &\leq M^2\overline{e} - \frac{(M\overline{e})^2}{n} \qquad (15.6) \\ &= M^2\left(\overline{e} - \frac{\overline{e}^2}{n}\right) \end{aligned}$$

Z (15.3), i (15.5) i (15.6) dostajemy

$$M^2\left(\overline{e} - \frac{\overline{e}^2}{n}\right) \leq \frac{M(M-1)}{2}d \ ,$$

po podzieleniu przez M dostajemy

$$M\left(\overline{e} - \frac{\overline{e}^2}{n}\right) \leq \frac{M-1}{2}d \ .$$

Rozwiązując to równanie dostajemy

$$\begin{aligned} M &\leq \frac{dn}{dn - 2n\overline{e} + 2\overline{e}^2} \\ &= \frac{2dn}{2dn - n^2 + n^2 - 4n\overline{e} + 4\overline{e}^2} \\ &= \frac{2dn}{-n(n-2d) + (n - 2\overline{e})^2} \\ &\leq \frac{2dn}{-n(n-2d) + (n-2e)^2} \end{aligned}$$

Teraz korzystamy z założenia, że

$$\frac{e}{n} < \frac{1}{2}\left(1 - \sqrt{1 - \frac{2d}{n}}\right)$$

Mnożąc obie strony przez n i przenosząc między stronami dostajemy

$$n - 2e > \sqrt{n(n-2d)}$$

czyli

$$(n-2e)^2 > n(n-2d)$$

Zatem $-n(n-2d) + (n-2e)^2 \geq 1$, ponieważ to są same liczby naturalne. I dlatego dostajemy

$$M \leq 2dn \quad \square$$

Chcemy teraz oszacować funkcję $J_q(\cdot)$.

Lemat 15.3. *Niech $q \geq 2$ będzie liczbą całkowitą i niech $0 \leq x \leq 1 - \frac{1}{q}$. Wtedy zachodzą następujące nierówności:*

$$J_q(x) \geq 1 - \sqrt{1-x} \geq \frac{x}{2} \ ,$$

przy czym druga nierówność jest ścisła dla $x > 0$.

Dowód zostawimy jako ćwiczenie.

Z Lemat wynika, że można podać wersję ograniczenia Johnsona, która nie zależy od rozmiaru alfabetu

Twierdzenie 15.4 (Ograniczenie Johnsona, wersja niezależna od rozmiaru alfabetu)**.** *Jeśli $e \leq n - \sqrt{n(n-d)}$, to każdy kod o odległości d jest $(e/n, qnd)$-dekodowalny do listy, dla wszystkich q.*

Ograniczenie Johnsona jest ścisłe w tym sensie, że są kody które dla $\rho > J_q(\delta)$ mają ponad wielomianowo wiele słów w odległości ρn.

15.2 Dekodowania do listy a zawartość informacyjna

Odwracamy pytanie: poprzednio mówiliśmy, że *każdy* kod można dekodować do listy (przy pewnych parametrach). Teraz pytamy o to, jak dobre kody istnieją, przy czym tradycyjnie chcemy powiązać błędy z zawartością informacyjną.

Twierdzenie 15.5. *Niech $q \geq 2$, $0 \leq \rho < 1 - \frac{1}{q}$ oraz $\epsilon > 0$. Dla dostatecznie dużego n dla kodów o długości kodu n zachodzą następujące warunki:*

1. *Jeśli $R \leq 1 - H_q(\rho) - \epsilon$ to istnieje kod $(\rho, \mathcal{O}(1/\epsilon))$-dekodowalny do listy.*
2. *Jeśli $R > 1 - H_q(\rho) + \epsilon$, każdy kod (ρ, L)-dekodowalny do listy ma $L \geq q^{\Omega(n)}$.*

Uwaga.

- Można dekodować do listy prawie $1 - H_q(\rho)$ (gdzie ρ jest frakcją błędów). i jest to ścisłe ograniczenie.
- To jest dużo lepiej, niż dla poprawiania jednego błędu: z ograniczenia Singletona $R + \delta \leq 1 + 1/n$ a potrafimy poprawić $\delta/2$ błędów, czyli (pomijając $1/n$) nie więcej niż frakcję $\rho = \frac{1-R}{2}$ błędów, czyli

$$R \leq 1 - 2\rho \ .$$

- Pokażemy dowód dla kodów nieliniowych, ale da się go porawić dla liniowych.

Zastosujemy metodę probabilistyczną: wybierzemy losowo kod i pokażemy, że spełnia on zadane własności z niezerowym prawdopodobieństwem. Dokładniej, pokażemy silniejszą wersję Twierdzenia 15.5: losowy kod z wysokim prawdopodobieństwem jest (ρ, L)-dekodowalny do listy, jeśli

$$R \leq 1 - H_q(\rho) - \frac{1}{L} \ .$$

Dla $\epsilon = \frac{1}{L}$ to daje Twierdzenie 15.5.

Twierdzenie 15.6. *Niech $q \geq 2$ będzie liczbą całkowitą, a $0 < \rho < 1 - \frac{1}{q}$ liczbą rzeczywistą.*

1. *Niech $L \geq 1$ będzie liczbą całkowitą, wtedy istnieje kod (ρ, L)-dekodowalny do listy o zawartości informacyjnej*
$$R \leq 1 - H_q(\rho) - \frac{1}{L}$$
2. *Dla każdego kodu (ρ, L)-dekodowalnego do listy o zawartości informacyjnej $1 - H_q(\rho) + \epsilon$, zachodzi $L \geq 2^{\Omega(\epsilon n)}$.*

Zauważmy, że L może być funkcją n.

Dowód. Zaczynamy od dowodu części pierwszej.

Wybierzmy losowo kod C rozmiaru q^k, gdzie $k \leq 1 - H_q(\rho) - 1/L$. Tj. dla każdej wiadomości losowo i niezależnie wybieramy $C(m)$.

Dla $\vec{y} \in \mathbb{F}_q^n$ oraz $\vec{m}_0, \ldots, \vec{m}_L \in \mathbb{F}_q^k$ definiują „złe zdarzenie" jako

$$C(\vec{m}_i) \in B(\vec{y}, \rho n)$$

dla wszystkich $0 \leq i \leq L$. Ustalmy takie $\vec{y} \in \mathbb{F}_q^n$ i $\vec{m}_0, \ldots, \vec{m}_L \in \mathbb{F}_q^k$.

Dla ustalonego i prawdopodobieństwo bycia w tej kulce to

$$\mathbb{P}[C(\vec{m}_i) \in B(y, \rho n)] = \frac{\mathrm{Vol}_q(\rho n, n)}{q^n} \leq q^{-n(1-H_q(\rho))},$$

Czyli prawdopodobieństwo wystąpienia złego zdarzenia wynosi

$$\mathbb{P}\left[\bigwedge_{i=0}^{L} C(\vec{m}_i) \in B(y, \rho n)\right] = \prod_{i=0}^{L} \mathbb{P}\left[C(\vec{m}_i) \in B(y, \rho n)\right] \leq q^{-n(L+1)(1-H_q(\rho))},$$

W takim razie szansa na złe zdarzenie to

$$\begin{aligned}
\mathbb{P}[\text{złe zdarzenie}] &\leq \underbrace{q^n}_{\text{wybór } \vec{y}} \underbrace{\binom{q^k}{L+1}}_{\text{wybór } L+1 \text{ słów kodowych}} q^{-n(L+1)(1-H_q(\rho))} \\
&\leq q^n q^{k(L+1)} q^{-n(L+1)(1-H_q(\rho))} \\
&\leq q^n q^{Rn(L+1)} q^{-n(L+1)(1-H_q(\rho))} \\
&= q^{-n(L+1)[1-H_q(\rho)-1/(L+1)-R]} \\
&\leq q^{-n(L+1)[1-H_q(\rho)-1/(L+1)-1+H_q(\rho)+1/L]} && \text{podstawiamy } R \leq 1 - H_q(\rho) - 1/L \\
&= q^{-n/L} \\
&< 1
\end{aligned}$$

Przechodzimy do części drugiej. Chcemy wykazać istnienie $\vec{y} \in \mathbb{F}_q^n$ tak, że $|C \cap B(y, \rho n)|$ jest wykładnicza (ze względu na n) dla każdego kodu C i zawartości informacyjnej $R \geq 1 - H_q(\rho) + \epsilon$.

Ponownie użyjemy metody probabilistycznej. Wylosujmy jednostajnie $\vec{y} \in \mathbb{F}_q^n$. Chcemy obliczyć, jaka jest oczekiwana liczba elementów kodu C w odległości ρn od $\vec{y}$. W tym celu oszacujemy najpierw prawdopodobieństwo dla jednego wektora; ustalmy więc $\vec{c} \in C$. Wtedy

$$\begin{aligned}
\mathbb{P}[\vec{c} \in B(\vec{y}, \rho n)] &= \mathbb{P}[\vec{y} \in B(\vec{c}, \rho n)] && \text{symetria} \\
&= \frac{\mathrm{Vol}_q(\rho n, n)}{q^n} && \vec{y} \text{ wybieramy losowo jednostajnie} \\
&\geq q^{-n(1-H_q(\rho)+o(1))} ,
\end{aligned}$$

Przechodząc do wartości oczekiwanej:

$$\begin{aligned}
\mathbb{E}[|C \cap B(y, \rho n)|] &= \sum_{\vec{c} \in C} \mathbb{E}[1_{\vec{c} \in B(\vec{y}, \rho n)}] \\
&= \sum_{\vec{c} \in C} \mathbb{P}[\vec{c} \in B(\vec{y}, \rho n)] \\
&\geq \sum_{\vec{c} \in C} q^{-n(1-H_q(\rho)+o(1))} \\
&= q^{nR-n(1-H_q(\rho)+o(1))} \\
&= q^{n(\overbrace{R-1+H_q(\rho)}^{\geq \epsilon}-o(1))} \\
&\geq q^{\Omega(\epsilon n)}
\end{aligned}$$

W takim razie istnieje $\vec{y}$, dla którego kulka zawiera wykładniczo wiele elementów kodu. □

Pozostaje parę pytań otwartych

- Czy istnieją jawnie zdefiniowane kody osiągające ograniczenie z Twierdzenia 15.5?
- Czy możemy osiągnąć wydajność dekodowania do listy przy użyciu wydajnych efektywnych algorytmów? [Tak, ale to trudne.]
- Czy możemy podać efektywny algorytm dekodowania listy, który osiągnie ograniczenie Johnsona? W szczególności, czy możemy efektywnie dekodować o zawartości informacyjnej R dla frakcji $1 - \sqrt{R}$ błędów?

Rozdział 16

Dekodowania do list kodów Reeda-Solomona

Na podstawie [8, Rozdział 15.2].

Z wersji ograniczenia Johnsona niezależnej od wielkości alfabetu mamy (Twierdzenie 15.4), że potrafimy dekodować do listy dla frakcji błędów

$$\rho \leq 1 - \sqrt{1-\delta}$$

gdzie $\delta = \frac{d}{n}$ jest odległością względną kodu. Dla kodów RS mamy $\delta = 1 - R + 1/n$, czyli $1 - \delta \approx R$.

W tym rozdziale podamy algorytm, który dekoduje do listy kody RS dla ; dokładniej, dla kodu o zawartości informacyjnej R dekoduje do frakcji $1 - \sqrt{R}$ błędów, czyli (dla dużych n) w przybliżeniu do wartości z ograniczenia Johnsona. Zauważmy też, że z ograniczenia Johnsona taki kod jest $(1 - \sqrt{R}, O(n^2))$- dekodowalny do listy, czyli wielkość wyjścia jest ograniczona.

Najpierw przeformułujemy trochę algorytm BW.

16.1 Sformułowanie

Rozważmy $[n, k]_q$ kod Reed-Solomona, który ewaluujemy w punktach $(\alpha_1, \ldots, \alpha_n)$. Chcemy rozwiązać następujący problem:

Wejście Słowo $\vec{y} = (y_1, \ldots, y_n), y_i \in \mathbb{F}_q$ i parametr błędu $e = n - t$.

Wyjście Wszystkie wielomiany $P(X) \in \mathbb{F}_q[X]$ stopnia $\leq k - 1$, takie że $P(\alpha_i) = y_i$ dla co najmniej t wartości i.

Przypomnijmy sobie algorytm BW dla przypadku $t > \frac{n-k}{2}$; przeformułowujmy go trochę, tak by był przydatny do dekodowania do listy.

Algorytm wyglądał następująco

Krok 1 Znajdź wielomiany $N(X)$ stopnia $k + e - 1$ oraz $E(X)$ stopnia e, takie że

$$N(\alpha_i) = y_i E(\alpha_i) \ .$$

Krok 2 Oblicz $P(X) = \frac{N(X)}{E(X)}$ i zwróć (zakładając, że jest dostatecznie blisko).

Kluczowe uogólnienie jest następujące: możemy potraktować α_i jako podstawienie pod kolejną zmienną (Y), czyli rozpatrujemy wielomian

$$Q(X, Y) = Y E(X) - N(X)$$

Wtedy $P(X) = \frac{N(X)}{E(X)}$ można odzyskać z Q: tak zdefiniowane P spełnia warunek $Y - P(X) | Q(X,Y)$:

$$E(X) \cdot \left(Y - \frac{N(X)}{E(X)} \right) = YE(X) - N(X) = Q(X) \ .$$

I odwrotnie, tj. $Y - P(X) | Q(X,Y)$ implikuje, że $P(X)$ jest wyjściem (o ile jest dostatecznie blisko): jeśli $R(X,Y)(Y - P(X)) = Q(X,Y) = YE(X) - N(X)$ to z tego, że Y występuje tylko w pierwszej potędze wnioskujemy, że $R(X,Y) = E(X)$ i dostajemy, że $E(X) | N(X)$.

Tak więc algorytm można przeformułować następująco:

Interpolacja Znajdź $Q(X,Y)$ spełniający

$$Q(\alpha_i, y_i) = 0 \quad \text{dla } 1 \leq i \leq n$$

Faktoryzacja Jeśli $(Y - P(X)) | Q(X,Y)$ to zwróć $P(X)$ (jeśli wektor wartości jest dostatecznie blisko).

Uwaga. Wiadomo, że można rozłożyć na czynniki wielomian dwóch zmiennych nad $\mathbb{F}_q$ w wielomianowym czasie. Nie będziemy tego pokazywać.

Będziemy rozważać uogólnienia tego podejścia, przy czym

- Trzeba zadbać, że mamy nie za dużo warunków, żeby takie Q istniało.
- Będziemy nakładać jakieś ograniczenia na $Q(X,Y)$, żeby to wszystko miało sens (np. BW nakłada warunek, że jest on postaci $YE(X) + N(X)$ dla pewnych wielomianów jednej zmiennej i ogranicza ich stopnie.)

16.2 Algorytm 1

Najważniejszą cechą algorytmu będzie staranne dobranie stopni wielomianu oraz właściwa definicja stopnia wielomianu dwóch zmiennych.

Przypomnijmy, że maksymalny zmiennej w wielomianie to maksymalna potęga X w którymkolwiek jednomianie, oznaczamy $\deg_X(Q(X,Y))$ oraz $\deg_Y(Q(X,Y))$.

Jeśli $\deg_X(Q) = a$ oraz $\deg_Y(Q) = b$, to możemy zapisać Q jako

$$Q(X,Y) = \sum_{\substack{0 \leq i \leq a \\ 0 \leq j \leq b}} c_{i,j} X^i Y^j \ ,$$

gdzie współczynniki $c_{i,j} \in \mathbb{F}_q$. Liczba współczynników jest wtedy równa $(a+1)(b+1)$. Pomysł jest następujący: ograniczymy $\deg_X(Q), \deg_Y(Q)$ tak, że będziemy mieli gwarancję, że Q istnieje.

Po pierwsze musimy zadbać o to, że Q w ogóle istnieje. Zauważmy, żę wystarczy, że liczba współczynników $Q(X,Y)$ (która wynosi $(\ell+1)(n/\ell+1)$) jest większa lub równa liczbie więzów (czyli wartości w punktach), których jest n: tak jest, bo wszystkie równania mają po prawej stronie 0 i w takim razie zbiór rozwiazań to jądro odpowiedniej macierzy, chodzi tylko o wymiar tego jądra (żeby nie było wymiaru 0). Na szczęście to zachodzi:

$$(n+1)(n/\ell + 1) > n \cdot n/\ell = n \ .$$

Musimy też pokazać, że jeśli P jest stopnia $\deg P < k$ oraz wektor wartości $(P(\alpha_i))_{i=1}^n$ jest w odległości najwyżej e od $\vec{y}$ (czyli ma przynajmniej $t = n - e$ wartości wspólnych) to $Y - P(X)$ dzieli $Q(X,Y)$, czyli nie pominiemy go przy wypisywaniu.

Zdefiniujemy

$$R(X) = Q(X, P(X)).$$

Wtedy $Y - P(X)$ dzieli $Q(X,Y)$ wtedy i tylko wtedy, gdy $R(X) \equiv 0$:

Algorytm 8 1. Algorytm dekodujący RS do listy (Sudan)

Założenie: $n \geq k \geq 1$, $\ell \geq 1$, $e = n - t$, n par $\{(\alpha_i, y_i)\}_{i=1}^{n}$ ▷ ℓ jest ustalone jako $\sqrt{n(k-1)}$

znajdź niezerowy wielomian $Q(X, Y)$ spełniający

- $\deg_X(Q) \leq \ell$
- $\deg_Y(Q) \leq \frac{n}{\ell}$
- $Q(\alpha_i, y_i) = 0$ dla $0 \leq i \leq n$

$L \leftarrow \emptyset$

for każdy dzielnik $Y - P(X)$ wielomianu Q **do**

  if $\mathrm{d_H}(\vec{y}, (P(\alpha_i))_{i=1}^{n}) \leq e$ oraz $\deg(P) < k$ **then**

5: $L \leftarrow L \cup \{P\}$

return L

Lemat 16.1.

$$Y - P(X) | Q(X, Y) \iff R(X) := Q(X, P(X)) \equiv 0$$

Dowód pozostawimy jako ćwiczenie (w prawo łatwo, w lewo trochę trudniej).

Czyli chcemy pokazać, że $R(X) \equiv 0$. W tym celu oszacujemy jego stopień oraz liczbę jego pierwiastków.

Policzmy stopień R:

$$\begin{aligned} \deg(R) &\leq \deg_X(Q) + \deg(P) \cdot \deg_Y(Q) \\ &\leq \ell + \frac{n(k-1)}{\ell} \end{aligned}$$

(Oszacowanie $\deg(P) \leq k - 1$ bierze się z tego, że to ma być wielomian do policzenia wartości w kodzie RS, czyli ma stopień $< k$).

Zauważmy, że to jest suma dwóch liczb o ustalonym iloczynie ($= n(k-1)$ i z nierówności między średnimi ta suma jest najmniejsza, gdy te liczby są równe, tj. dla $\ell = \frac{n(k-1)}{\ell}$, czyli dla $\ell = \sqrt{n(k-1)}$, i wynosi wtedy

$$\deg(R) \leq 2\sqrt{n(k-1)} \ .$$

Z drugiej strony, jeśli $P(\alpha_i) = y_i$, co jest prawdą dla przynajmniej t różnych par (α_i, y_i), to

$$Q(\alpha_i, y_i) = Q(\alpha_i, P(\alpha_i)) = 0.$$

Zatem α_i jest pierwiastkiem $R(X)$ dla t różnych indeksów i, czyli jeśli R jest niezerowy, to ma co najmniej t pierwiastków. To daje

$$t \leq 2\sqrt{n(k-1)}$$

Czyli dla *większych* t, tj.:

$$t > 2\sqrt{n(k-1)}$$

algorytm poprawnie dekoduje do listy.

Twierdzenie 16.2. *Algorytm dekoduje do listy kody Reeda-Solomona o zawartości informacyjnej R z frakcji $1-2\sqrt{R}$ błędów. Algorytm ten może być zaimplementowany w czasie wielomianowym.*

Dowód. Pozostaje oszacować rozmiar frakcji: potrafimy zdekodować do listy dla $n-t$ błędów dla $t > 2\sqrt{n(k-1)}$. Frakcją błędów $1-2\sqrt{R}$ oznacza

$$\begin{aligned} 1-2\sqrt{R} &\geq \frac{e}{n} \\ &= 1-\frac{t}{n} \end{aligned}$$

czyli

$$t \geq 2n\sqrt{R}$$

Przypominając sobie, że $R = \frac{k}{n}$, dostajemy

$$\begin{aligned} t &\geq 2\sqrt{kn} \\ &> 2\sqrt{n(k-1)} \end{aligned}$$

czyli założenie na t jest spełnione.

Interpolacja może być zaimplementowana w czasie wielomianowym przy użyciu eliminacji Gauba. Faktoryzację robimy zewnętrznym algorytmem. □

Ograniczenie $1-2\sqrt{R}$ jest istotnie lepsza, niż dla klasycznego poprawiania błędów ($\frac{1-R}{2}$), Jest to jednak daleko od frakcji $1-\sqrt{R}$ z ograniczenia Johnsona.

16.3 Drugie podejście

Będziemy poprawiać oszacowanie na stopień wielomianu: zauważmy, że z jednej strony nakładaliśmy warunki na $\deg_X$ oraz $\deg_Y$ a potem maksymalny stopień szacowaliśmy przez $\deg_X(Q) + (k-1)\deg_Y(Q)$. Ale maksymalne $\deg_X$ oraz $\deg_Y$ mogą nie być w jednym jednomianie, możemy też takie wystąpienia ograniczyć. Co więcej, wiemy, przez co będziemy mnożyć $\deg_X$ a przez co $\deg_Y$.

Definicja 16.3 (Stopień ważony)**.** Dla jednomianu $P = X^kY^\ell$ dwóch zmiennych X, Y zdefiniujmy

$$\deg_{(x,y)}(X^kY^\ell) := xk + y\ell$$

Dla wielomianu Q jego $\deg_{(x,y)}(Q)$ definiujemy jako maksimum po jednomianach Q.

Uwaga. Zauważmy, że $\deg_X = \deg_{(1,0)}$, $\deg_Y = \deg_{(0,1)}$ oraz $\deg = \deg_{(1,1)}$.

Lemat 16.4. *Niech $Q(X,Y)$ będzie wielomianem dwóch zmiennych oraz $\deg_{(1,w)}(Q) = D$. Niech $P(X)$ będzie wielomianem, takim że $\deg(P) \leq w$. Wtedy*

$$\deg(Q(X, P(X))) \leq D \ .$$

Prosty dowód pozostawiamy jako ćwiczenie.

Zauważmy, że wielomian dwóch zmiennych $Q(X,Y)$, gdzie $\deg_{(1,w)}(Q) \leq D$ można przedstawić w postaci:

$$Q(X,Y) = \sum_{\substack{i,j\geq 0 \\ i+wj\leq D}} c_{i,j}X^iY^j \ ,$$

gdzie $c_{i,j} \in \mathbb{F}_q$.

Nasz nowy algorytm jest zasadniczo taki sam jak poprzedni, z tym że przy interpolacji obliczamy wielomian dwóch zmiennych spełniających warunek na $\deg_{(1,k-1)}(Q)$.

Analogicznie jak w dowodzie poprawności poprzedniego algorytmu, musimy pokazać że

Algorytm 9 2. Algorytm dekodujący RS do listy (Algorytm Sudana)

Założenie: $n \geq k \geq 1$, $D \geq 1$, $e = n - t$, n par $\{(\alpha_i, y_i)\}_{i=1}^{n}$

▷ D jest ustalone jako $D = \sqrt{2n(k-1)}$

znajdź niezerowy wielomian $Q(X, Y)$ spełniający

- $\deg_{(1,k-1)}(Q) \leq D$
- $Q(\alpha_i, y_i) = 0$ dla $0 \leq i \leq n$

$L \leftarrow \emptyset$

for każdy dzielnik $Y - P(X)$ wielomianu Q **do**

5: **if** $\mathrm{d_H}(\vec{y}, (P(\alpha_i))_{i=1}^{n}) \leq e = n - t$ oraz $\deg(P) < k$ **then**

$L \leftarrow L \cup \{P\}$

return L

- D jest dostatecznie duże, tak aby liczba współczynników Q była większa niż n;
- znaleźć możliwie małe t, takie że jeśli $P(\alpha_i) = y_i$ dla przynajmniej t różnych wartości i, to $Y - P(X) | Q(X, Y)$.

Drugi punkt jest prosty: z Lematu 16.4 mamy, że $\deg_{(1,k-1)} Q(X, Y) \leq D$ implikuje, że $\deg(R) \leq D$. Tak jak poprzednio, jeśli wybierzemy $t > D$, to pokaże to, że $R \equiv 0$. W takim razie chcemy policzyć, jakie jest najmniejsze możliwe D, dla którego istnieje niezerowe Q.

Zdefiniujmy

$$N_{k,D} = |\{(i, j) \ : \ i + (k-1)j \leq D, i, j \in \mathbb{N}\}| \ .$$

Można policzyć (ćwiczenie), że

$$\begin{aligned} N_{k,D} &= \sum_{j=0}^{\lfloor \frac{D}{k-1} \rfloor} \sum_{i=0}^{D-j(k-1)} 1 \\ &\vdots \\ &> \frac{D(D+2)}{2(k-1)} \end{aligned}$$

Czyli uda się zinterpolować, jeśli

$$\begin{aligned} \frac{D(D+2)}{2(k-1)} &\geq n && \iff \\ (D+1)^2 &\geq 2(k-1)n + 1 && \iff \\ D &\geq \sqrt{2(k-1)n+1} - 1 \end{aligned}$$

Wystarczy więc

$$D = \left\lceil \sqrt{2n(k-1)+1} - 1 \right\rceil$$

Tak więc wystarczy założyć, że

$$t = \left\lceil \sqrt{2(k-1)n+1} \right\rceil \ .$$

Twierdzenie 16.5. *Algorytm 2 może dekodować do listy kody Reeda-Solomona o zawartości informacyjnej R do frakcji $\leq 1 - \sqrt{2R}$ błędów.*

Dowód jest analogiczny, jak poprzednio.

Dowód. Wystarczy pokazać, że dla frakcji błędów $1 - \sqrt{2R}$ zachodzi ograniczenie

$$t = n - e \geq \sqrt{2(k-1)n + 1} \ .$$

Po podzieleniu przez n mamy

$$t/n \geq \sqrt{2(k-1)/n + 1/n^2} \ .$$

Z założeń

$$\begin{aligned} 1 - \rho &\geq \sqrt{2R} && \text{założenie} \\ &= \sqrt{2k/n} \\ &> \sqrt{2(k-1)/n + 1/n^2} && \square \end{aligned}$$

16.4 Podejście trzecie

Podamy algorytm dekodowania do listy dla kodów Reed-Solomona, który działa dla frakcji do $1 - \sqrt{R}$ błędów. Główną ideą jest zapewnienie, że $Y - P(X)$ jest pierwiastkiem r-krotnym. To dodaje ograniczeń na współczynniki (czyli musimy ich mieć więcej), ale pozwala lepiej oszacować stopień R (bo pierwiastki są wielokrotne).

Potrzebujemy jakiejś definicji pierwiastka wielokrotnego: intuicja (dla punktu $0, 0$): punkt $(0, 0)$ jest pierwiastkiem r-krotnym Q, jeśli Q można przedstawić jako iloczyn $Q = Q_1 \cdots Q_r$, taki że $Q_i(0, 0) = 0$. Zauważmy, że wtedy Q_i nie ma wyrazu wolnego, tj. każdy wyraz ma stopień przynajmniej 1. Czyli Q nie ma wyrazu stopnia mniejszego niż r (i to są te dodatkowe więzy)

Definicja 16.6 (wielokrotny pierwiastek)**.** Mówimy, że $(0, 0)$ jest pierwiastkiem r-krotnym Q, jeśli Q nie ma jednomianu stopnia mniejszego niż r.

W ogólności (α, β) jest pierwiastkiem r-krotnym Q, jeśli $(0, 0)$ jest pierwiastkiem r-krotnym wielomianu

$$Q_{\alpha,\beta}(X, Y) = Q(X + \alpha, Y + \beta) \ .$$

Algorytm 10 3. Algorytm dekodujący RS do listy (Guruswani-Sudan)

Założenie: $n \geq k \geq 1$, $D \geq 1$, $r \geq 1$, $e = n - t$, n par $\{(\alpha_i, y_i)\}_{i=1}^n$

▷ r jest ustalone jako $2(k-1)n$, zaś D jako $\sqrt{n(k-1)r(r+1)}$

znajdź niezerowy wielomian $Q(X, Y)$ spełniający

- $\deg_{(1,k-1)}(Q) \leq D$
- (α_i, y_i) jest r-krotnym pierwiastkiem Q

$L \leftarrow \emptyset$

for każdy dzielnik $Y - P(X)$ wielomianu Q **do**

5: **if** $\mathrm{d_H}(\vec{y}, (P(\alpha_i))_{i=1}^n) \leq e$ oraz $\deg(P) < k$ **then**

$L \leftarrow L \cup \{P\}$

return L

Lemat 16.7. *Ograniczenia na pierwiastki dają* $\frac{r(r-1)}{2}$ *więzów do interpolacji dla każdej pary* (α_i, y_i).

Dowód za chwilę.

Lemat 16.8. *Niech* $Q(X,Y)$ *będzie wielomianem obliczonym przez Algorytm. Niech* $P(X)$ *będzie wielomianem stopnia* $\leq k-1$*, takim że* $P(\alpha_i) = y_i$ *dla conajmniej* $t > D/r$ *różnych wartości* i*. Wtedy* $(Y - P(X))|Q(X,Y)$.

Dowód za chwilę.

Żeby pokazać poprawność algorytmu, będziemy musieli pokazać, że

$$\frac{D(D+2)}{2(k-1)} \geq n\frac{(r+1)r}{2}$$

(lewa strona to liczba współczynników, zaś prawa: liczba więzów). Bierzemy więc

$$D = \left\lceil \sqrt{(k-1)nr(r+1)+1} \right\rceil - 1 \ .$$

Z Lematu 16.8 musimy jeszcze zapewnić $t > D/r$. Czyli chcemy

$$t > \frac{\left\lceil \sqrt{(k-1)nr(r+1)+1} \right\rceil - 1}{r}$$

Wystarczy więc t spełniające

$$t \geq \frac{\left\lceil \sqrt{(k-1)nr(r+1)+1} \right\rceil}{r}$$

Ponieważ

$$\lceil x/r \rceil \geq \lceil x \rceil / r$$

To wystarczy nam

$$t \geq \left\lceil \sqrt{(k-1)n(1+1/r)+1/r^2} \right\rceil$$

Pozostaje nam ustalić r, bierzemy

$$r = 2(k-1)n$$

co daje

$$\begin{aligned} t &\geq \left\lceil \sqrt{(k-1)n + \underbrace{1/2 + 1/(4(k-1)^2n^2)}_{<1}} \right\rceil \\ &= \left\lceil \sqrt{(k-1)n+1} \right\rceil \end{aligned}$$

Zauważmy też, że skoro t zawsze jest całkowite, to wystarczy, że pokażemy, że

$$t \geq \sqrt{(k-1)n+1}$$

bo jako liczba całkowita spełnia też zaokrąglenie w górę.

Twierdzenie 16.9. *Algorytm może dekodować do listy kody Reeda-Solomona o współczynniku R do frakcji (prawie) $1 - \sqrt{R}$ błędów.*

Dowód. Szacujemy podobnie jak poprzednio, dostajemy (korzystając z $R = \frac{k}{n}$), że

$$t \geq \sqrt{kn} > \sqrt{(k-1)n+1}$$

Ponieważ t jest całkowite, to spełnia też ograniczenie z sufitem. □

Dowód Lematu 16.7. Niech

$$\begin{aligned} Q(X,Y) &= \sum_{\substack{i,j\geq 0 \\ i+(k-1)j\leq D}} c_{i,j} X^i Y^j \qquad \text{oraz} \\ Q_{\alpha,\beta}(X,Y) &= Q(X+\alpha, Y+\beta) \\ &= \sum_{i,j} c_{i,j}^{\alpha,\beta} X^i Y^j \end{aligned}$$

Z definicji wielokrotnego pierwiastka, $Q_{\alpha,\beta}(X,Y)$ nie ma jednomianu stopnia $< r$. Czyli musimy nałożyć więzy

$$c_{i,j}^{\alpha,\beta} = 0$$

dla $i,j : 0 \leq i+j < r$. Jest ich

$$\underbrace{r}_{i=0} + \underbrace{(r-1)}_{i=1} + \cdots + 1 = (r+1)r/2$$

Jak te więzy przekładają się na wiezy na oryginalne współczynniki, w szcególności, czy są żądanej postaci? Rozwińmy definicję $Q_{\alpha,\beta}$:

$$Q_{\alpha,\beta}(X,Y) = \sum_{i,j} c_{i,j}^{\alpha,\beta} X^i Y^j = \sum_{\substack{i',j' \\ i'+(k-1)j'\leq D}} c_{i',j'} (X+\alpha)^i (Y+\beta)^j$$

Porównując współczynniki przy $X^i Y^j$ dla obu stron dostajemy

$$c_{i,j}^{\alpha,\beta} = \sum_{\substack{i'\geq i \\ j'\geq j \\ i'+(k-1)j'\leq D}} c_{i',j'} \binom{i'}{i} \binom{j'}{j} \alpha^{i'-i} \beta^{j'-j} \ .$$

I to na kombinacje jak po prawej stronie będziemy nakładać więzy. □

Pozostało pokazać Lemat 16.8.

dowód Lematu 16.8. Niech ponownie

$$R(X) = Q(X, P(X)) \ .$$

Pokazaliśmy już wcześniej, że

$$(Y - P(X)) | Q(X,Y) \iff R(X) \equiv 0 \ .$$

Chcemy więc pokazać, że $R(X) \equiv 0$.

Skoro $\deg P \leq k-1$ to $\deg(R) \leq \deg_{(1,k-1)}(Q) \leq D$ (z konstrukcji). Czyli wystarczy pokazać, że R ma przynajmniej $tr > D$ pierwiastków, uwzględniając krotność, co pokazujemy w poniższym Lemacie. □

Lemat 16.10. *Jeśli $P(\alpha_i) = y_i$ to $(X - \alpha_i)^r | R(X)$, tj. α_i jest r-krotnym pierwiastkiem R.*

Dowód. Zdefiniujmy

$$R_{\alpha_i,y_i}(X) = R(X + \alpha_i)$$

Zauważmy, że z definicji $R_{\alpha_i,y_i}(X)$ mamy, że

$$(X - \alpha_i)^r | R \iff X^r | R_{\alpha_i,y_i}(X)$$

Pokażemy zatem, że $X^r | R_{\alpha_i,y_i}(X)$.

$$\begin{aligned} R_{\alpha_i,y_i}(X) &= R(X + \alpha_i) \\ &= Q(X + \alpha_i, P(X + \alpha_i)) \end{aligned}$$

Analogicznie zdefiniujym przesunięte P:

$$P_{\alpha_i,y_i}(X) = P(X + \alpha_i) - y_i$$

Wtedy

$$\begin{aligned} R_{\alpha_i,y_i}(X) &= Q(X + \alpha_i, P(X + \alpha_i)) \\ &= Q(X + \alpha_i, P_{\alpha_i,y_i}(X) + y_i) \\ &= Q_{\alpha_i,y_i}(X, P_{\alpha_i,y_i}(X)) \end{aligned}$$

Dla dobrego indeksu i mamy $P(\alpha_i) = y_i$ i w takim razie

$$P_{\alpha_i,y_i}(0) = 0 \ .$$

Czyli X jest dzielnikiem $P_{\alpha_i,y_i}(X)$, czyli

$$P_{\alpha_i,y_i}(X) = X \cdot g_i(X)$$

dla jakiegoś wielomianu $g(X)$. Wtedy

$$\begin{aligned} R_{\alpha_i,y_i}(X) &= \sum_{i',j'} c_{i',j'}^{\alpha_i,y_i} X^{i'} (P_{\alpha_i,y_i}(X))^{j'} \\ &= \sum_{i',j'} c_{i',j'}^{\alpha_i,y_i} X^{i'} (Xg(X))^{j'} \\ &= \sum_{i',j'} c_{i',j'}^{\alpha_i,y_i} X^{i'+j'} g_i(X)^{j'} \end{aligned}$$

Skoro $Q_{\alpha_i,y_i}(X, Y)$ nie ma jednomianu stopnia $< r$ to $c_{i',j'} = 0$ dla $i' + j' < r$. Tak więc $X^r | R_{\alpha_i,y_i}(X)$. □

Rozdział 17

Kody lokalnie korekcyjne

Na podstawie [9, Rozdział 15].

Scenariusz wygląda następująco: myślimy, że długość słowa kodowego jest bardzo duża, my zaś chcemy odczytać tylko parę bitów. Może dlatego, że dłuższe słowa kodowe są ogólnie lepsze a chcemy symulować krótsze; ale też to podejście jest istotne z punktu widzenia złożoności obliczeniowej (słowo kodowe reprezentuje np. wszystkie wartościowania funkcji, a my chcemy wartość w jednym konkretnym punkcie). Chcemy więc jeden element ze słowa kodowego, ale czytając podliniowe wiele (w najlepszym przypadku: $\mathcal{O}(1)$, w najgorszym powiedzmy $\mathcal{O}(n^\epsilon)$) wiele elementów słowa kodowego. Nie da się tego zrobić deterministycznie (dla rozsądnych parametrów kodu): być może zepsute są wszystkie elementy słowa kodowego, które obejrzymy. Rozwiązaniem jest wylosowanie elementów, które chcemy obejrzeć.

Definicja 17.1 (Kody lokalnie korekcyjne)**.** $C \leq \mathbb{F}_q^n$ jest (δ, Q, γ) kodem lokalnie korekcyjnym, jeśli istnieje zrandomizowany algorytm A który dla każdego słowa w takiego że $\mathrm{d_H}(w, c) \leq \delta n$ dla pewnego $c \in C$ oraz każdej pozycji i po zadaniu najwyżej Q pytań o elementy w zwraca z prawdopodobieństwem błędu najwyżej γ element $c[i]$.

Uwaga. A ma dostęp do wyroczni dla w, dla dowolnego j ma w stałym czasie dostęp do $w[j]$.

Zauważmy, że dla kodów binarnych definicja ma sens dla $\gamma < \frac{1}{2}$, bo dla większego prawdopodobieństwa błędy możemy po prostu wylosować wartość.

Uwaga. Są też inne, nierównoważne, ale zbliżone ideą definicje:

- chcemy odtworzyć element oryginalnej informacji
- chcemy rozpoznać, czy to jest słowo kodowe
- ...

Na razie zajmiemy się problemem, czy takie kody w ogóle istnieją i jakie są mnie więcej zależności parametrów.

Q	n (jako funkcja k)	Uwagi
2	$n = \Theta(2^k)$	
$\mathcal{O}(\log n)$	$k \leq n \leq poly(n)$	
$\mathcal{O}(n^\epsilon)$	$k \leq n \leq (1+\alpha)n$	(dowolnie małe α)

Wyniki z powyższej tabeli zaprezentujemy dla kodów RM (nad alfabetem binarnym).

17.1 Przykład wstępny: kod Hadamarda ($Q = 2$)

Dla kodu Hadamarda, przypomnijmy, że dla danego k mamy $n = 2^k$ i słowo kodowe to zapis kolejnych iloczynów iloczynów skalarnych $w \in 2^k$ ze wszystkimi słowami w $\mathbb{F}_2^k$.

Zauważmy, że słowa kodowe możemy potraktować jako funkcje $f_\alpha : \mathbb{F}_2^k \to \mathbb{F}_2$, która argumentowi β przypisuje iloczyn skalarny α i β:

$$f_\alpha(\beta) = \langle \alpha, \beta \rangle$$

Funkcje te są liniowe.

Dla argumentu α i słowa f wylosuj $\beta \in \mathbb{F}_2^k$ i zwróć $f(\beta) + f(\alpha + \beta)$

Algorytm 11 Algorytm lokalny dla kodu Hadamarda

Założenie: punkt $\alpha \in \mathbb{F}_2^k$, wyrocznia do $f : \mathbb{F}_2^k \to \mathbb{F}_2$, t. że $\mathrm{d_H}(f, c) < \delta 2^k$ dla pewnego c w kodzie Hadamarda
wylosuj jednorodnie $\beta \in \mathbb{F}_2^k$
return $f(\alpha + \beta) + f(\beta)$

Fakt 17.2. *Jeśli dla słowa kodowego c zachodzi*

$$\begin{aligned} c(\beta + \alpha) &= f(\beta + \alpha) \\ c(\beta) &= f(\beta) \end{aligned}$$

to wtedy również

$$c(\alpha) = f(\alpha + \beta) + f(\beta)$$

Prosty dowód wynika wprost z liniowości funkcji c.

W takim razie

$$\begin{aligned} \mathbb{P}[f(\alpha + \beta) + f(\beta) \neq c(\alpha)] &\leq \mathbb{P}[f(\alpha) \neq c(\alpha) \vee f(\beta) \neq c(\beta)] \\ &\leq \mathbb{P}[f(\alpha) \neq c(\alpha)] + \mathbb{P}[f(\beta) \neq c(\beta)] \\ &\leq 2\delta \end{aligned}$$

Wniosek 17.3. Kod Hadamarda jest $(2, \delta, 2\delta)$ kodem lokalnie korygującym.

Zauważmy, że choć dopuszczamy dowolną δ, to zgodnie z wcześniejszymi rozważaniami dla $\delta \geq \frac{1}{2}$ ten fakt nie mówi nic ciekawego.

17.2 Ogólniej

Teraz chcemy zrobić to samo dla $\mathcal{O}(\log n)$ zapytań.

Niestety, nie ma co liczyć, że nie trafimy w nic złego — oczekujemy (w sensie wartości oczekiwanej), że będzie $\delta \log n$ błędnych elementów, które trzeba jakoś poprawić.

Będziemy to robić dla kodów RM, weźmy na razie przypadek 2 zmiennych, ogólnie jest podobnie.

Niech nasze słowo kodowe to ewaluacja wielomianu (w naszyhc przykładach zawsze będziemy mieć $r \leq q - 2$)

$$F(X, Y) = \sum_{\substack{0 \leq i,j \\ i+j \leq r}} f_{i,j} X^i Y^j$$

Możemy sobie rozpisać na kwadracie rozmiaru $q \times q$. Chcemy jakąś dobrze zdefiniowaną część, która będzie miała liniowe własności.

Np. jeśli podstawimy $X \leftarrow 0$ to otrzymamy wielomian jednej zmiennej

$$F_{(0,1)}(Y) = F(0, Y) = \sum_{0 \le j < \min(q-1, r+1)} c_{0,j} Y^j$$

Analogicznie jest dla przypadku, gdy za Y podstawimy 0.

Linie przechodzące przez $(0, 0)$ mają pewne zalety:

- jest jedynie $\log q$ punktów na jednej linii
- dwie takie linie przecinają się tylko w $(0, 0)$

Oczywiście nie ma nic specjalnego w $(0, 0)$, zamiast tego można wziąć dowolny ustalony punkt.

Dla tego kwadratu odpowiada to wzięciu linii (poziomej lub pionowej) przechodzącej przez $(0, 0)$.

Ale cięcia wzdłuż dowolnej linii to słowa kodowe kodu RS: umiemy poprawić, o ile nie ma za dużo błędów. Dla ustalonej linii (poziomej lub pionowej) to nie daje nic ciekawego (błędy mogą być akurat tam). Ale zamiast tego możemy wziąć *dowolną prostą* przechodzącą przez $(0, 0)$ i wartościowanie na niej to dalej będzie kod RS.

Rozważmy podstawienie

$$L(Z) = (a_1 Z + b_1, a_2 Z + b_2)$$

Możemy myśleć, że to linia (gdy Z przebiega $\mathbb{F}_q$). Wtedy

$$F(L(Z)) = F(a_1 Z + b_1, a_2 Z + b_2) = \sum_{\substack{0 \le i,j < q-1 \\ i+j \le r}} f_{i,j} (a_1 Z + b_1)^i (a_2 Z + b_2)^j$$

jest wielomianem stopnia $\le \min(q - 2, r)$

Algorytm 12 Algorytm lokalny dla kodów $\mathrm{RM}(2, r)$

Założenie: punkt $(\alpha, \beta) \in \mathbb{F}_q^2$, wyrocznia do $G : \mathbb{F}_q^2 \to \mathbb{F}_q$, t. że $\mathrm{d_H}(G, F) < \delta q^2$ dla pewnego F w kodzie $\mathrm{RM}(2, r)$

wylosuj jednorodnie $(\sigma, \tau) \in \mathbb{F}_q^2 \setminus (0, 0)$

niech $L(Z) \leftarrow (\sigma Z + \alpha, \tau Z + \beta)$

odczytaj $G(L(\lambda))$ dla $\lambda \in \mathbb{F}_q$

$\widetilde{H}(Z) \leftarrow G(L(Z))$

popraw błędy w $\widetilde{H}$ do kodu RS H, $\deg(H) \le r$

return $H(0)$

Lemat 17.4. *Algorytm jest poprawny z prawdopodobieństwem* $\ge 1 - \frac{2\delta q}{q-r-1}$

Dowód. Obliczmy najpierw parametry kodu RS uzyskanego jako ograniczenie kody RM do linii: $n = q$ (tyle słów na linii), $k = r + 1$ (stopień), czyli poprawiamy do $< \left\lfloor \frac{q-r}{2} \right\rfloor$ błędów. Czyli algorytm dekodujący zwróci poprawne $H = F(L(Z))$ pod warunkiem, że jest mniej niż $\frac{q-r}{2}$ błędów.

Tj.

$$|\{\lambda \in \mathbb{F}_q \,:\, F(L(\lambda)) \neq G(L(\lambda))\}| < \left\lfloor \frac{q-r}{2} \right\rfloor$$

Dla każdej pozycji prawdopodobieństwo tego, że jest błędem, to $< 1/\delta$, czyli oczekiwana liczba błędów to

$$\mathbb{E}\left[\{\lambda \in \mathbb{F}_q \,:\, F(L(\lambda)) \neq G(L(\lambda))\}\right] < \delta q$$

Z nierówności Markowa

$$\begin{aligned}\mathbb{P}\left[\{\lambda \in \mathbb{F}_q \,:\, F(L(\lambda)) \neq G(L(\lambda))\} \geq \left\lfloor \frac{q-r}{2} \right\rfloor\right] &< \frac{\delta q}{\left\lfloor \frac{q-r}{2} \right\rfloor} \\ &\leq \frac{2\delta q}{q-r-1}\end{aligned}$$ □

Zauważmy, że podstawiając konkretną wartość r, np $r = q/2$ dostajemy $\approx 4\delta$ i dla $\delta < 1/4$ mamy dodatnie prawdopodobieństwo i $Q = q = \sqrt{n}$ zapytań.

17.3 Uogólnienia

Dla większych (ale stałych) m mamy $n = q^m$, możemy zrobić dokładnie tak samo, tu parametry to będą $Q = q = n^{1/m}$.

Dla niestałych m możemy wziąć $m = q/\log q$. To daje $n = q^{q/\log q} = 2^q$ i w takim razie $Q = q = \log(n)$.

17.4 Kody lokalnie korygujące o dużej zawartości informacyjnej

Okazuje się, że dla $q \approx (1+\epsilon)r$ istnieją wielomiany

$$f : \mathbb{F}_q^m \to \mathbb{F}_q$$

stopnia $> r$ takie że przecięcie do każdej linii $L : \mathbb{F}_q \to \mathbb{F}_q^m$ daje wielomian stopnia $\leq r$ (ale już dla np. $q > 2r$ nie ma takich wielomianów). Jako kod bierzemy wartościowanie takich wielomianów we wszystkich punktach. (Lifted Code).

Rozdział 18

Property Testing ≈ Locally testable codes

Na podstawie [7]

18.1 Property testing: Kody Hadamarda

Chcemy pokazać, że kod Hadamarda jest lokalnie testowalny, tj. dokładniej, że użycie $\mathcal{O}(1)$ zapytań (i losowości) pozwala stwierdzić (z dodatnim prawdopodobieństwem), czy dane słowo jest słowem kodu Hadamarda.

Definicja 18.1 (Kody lokalnie testowalne)**.** $C \leq \mathbb{F}_q^n$ jest (δ, Q, γ) kodem lokalnie testowalnym, jeśli istnieje zrandomizowany algorytm A który dla każdego słowa w takiego że $\mathrm{d_H}(w, C) \geq \delta n$ po zadaniu najwyżej Q pytań o elementy w odpowiada, czy $w \in C$, przy czym:

- jeśli $w \in C$ to algorytm zawsze odpowiada dobrze;
- jeśli $w \notin C$ to z prawdopodobieństwem przynajmniej γ odpowiada dobrze (czyli „$w \notin C$").

Zauważmy, że zbiór wszystkich słów kodowych kodu Hadamarda można utożsamić że zbiorem wszystkich homomorfizmów

$$\mathcal{F} = \{f : \mathbb{F}_2^n \to \mathbb{F}_2 \ : \ f \text{ jest liniowe}\}$$

Więc w zasadzie równoważnie chcemy stwierdzić, czy dane na wejściu słowo zadaje funkcję liniową.

Algorytm 13 Algorytm lokalny do sprawdzania, czy dana na wejściu funkcja jest homomorfizmem

Założenie: na wejściu dana jest (jako wyrocznia) funkcja $f : \mathbb{F}_2^n \to \mathbb{F}_2$ Chcemy sprawdzić, czy jest to homomorfizm.

```
wylosuj jednorodnie x, y ∈ F_2^n
if f(x) + f(y) = f(x + y) then
    akceptuj
else
    odrzuć
```

Uwaga, metody będą lekko probabilistyczne, lepiej jest przedefiniować odległość tak, by była z zakresu $[0, 1]$:

Definicja 18.2. Dla dwóch funkcji $f, g : \mathbb{F}_2^n \to \mathbb{F}_2$ definiujemy ich odległość $\delta(f, g)$ jako

$$\delta(f, g) := \mathbb{P}_{x \in \mathbb{F}_2^n}[f(x) \neq g(x)] = \frac{|\{x \in \mathbb{F}_2^n \ : \ f(x) \neq g(x)\}|}{2^n} \ .$$

Odległość rozszerzamy na odległość od zbioru w naturalny sposób (jako infimum po zbiorze).

Twierdzenie 18.3. *Jeżeli test się nie powiedziecie z prawdopodobieństwem $\delta_0 < \frac{2}{9}$, to*

$$\delta(f, \mathcal{F}) \leq 2\delta_0 \ .$$

Zauważmy, że to jest to sformułowanie, które chcemy: test nie powiedzie z prawdopodobieństwem $\geq 1/2$ odległości f od homomorfizmu.

Dowód. Ustalmy φ—bliskie f, powiedzmy

$$\delta(f, \varphi) \leq \epsilon$$

Zauważmy, że dla każdego $x, y \in \mathbb{F}_2^n$ zachodzi

$$\varphi(x) = \varphi(x + y) - \varphi(y)$$

Zauważmy, że

$$\begin{aligned} \mathbb{P}_y[\varphi(y) = f(y)] &\geq 1 - \epsilon \\ \mathbb{P}_y[\varphi(x + y) = f(x + y)] &\geq 1 - \epsilon \end{aligned}$$

a więc

$$\begin{aligned} \mathbb{P}_y[\varphi(x) = f(x + y) - f(y)] &\geq \mathbb{P}_y[f(x + y) = \varphi(x + y) \wedge f(y) = \varphi(y)] \\ &\geq 1 - 2\epsilon \ . \end{aligned}$$

Czyli z dużym prawdopodobieństwem (po losowaniu y) poprawna wartość $\varphi(x)$ wynosi $f(x + y) - f(y)$. Czyli chcemy zdefiniować $\varphi(x)$ jako najczęstszą wartość $f(x + y) - f(y)$.

Uwaga. To jest w pewnym sensie odpowiednik majority logic decoding.

Pokażmy, że przy założeniach z Twierdzenia 18.3, mamy

$$\delta(f, \varphi) \leq 2\delta_0$$

Zdefiniujmy zbiór złych x jako:

$$B = \{x \ : \ \mathbb{P}_y[f(x) \neq f(x + y) + f(y)] \geq \frac{1}{2}\}$$

Zauważmy, że jeśli $x \notin B$ to

$$\mathbb{P}_y[f(x) \neq f(x - y) - f(y)] > \frac{1}{2}$$

i w takim razie $\varphi(x) = f(x)$. Czyli $\delta(f, \varphi) \leq |B|/2^n$.

Jedyne co wiemy o f, to że prawdopodobieństwo, że zostanie ona odrzucona przez Algorytm jest małe. Wykorzystajmy to

$$\begin{aligned} \delta_0 &\geq \mathbb{P}_{x,y}[f(x + y) \neq f(x) + f(y)] \\ &\geq \mathbb{P}_{x,y}[f(x + y) \neq f(x) + f(y) \wedge x \in B] \\ &= \mathbb{P}_{x,y}[f(x + y) \neq f(x) + f(y) \,|\, x \in B] \cdot \mathbb{P}[x \in B] \end{aligned}$$

Przypomnijmy, że B to zbiór tych x, dla których lewe prawdopodobieństwo to przynajmniej $1/2$, czyli szacując

$$\delta_0 \geq \frac{1}{2} \cdot \mathbb{P}[x \in B]$$

Z czego dostajemy, że

$$\cdot\mathbb{P}[x \in B] \leq 2\delta_0$$

I w takim razie

$$\delta(f, \varphi) \leq 2\delta_0 \ . \qquad \square$$

Pozostały dowód opiera się na trzech lematach, dowody których pozostawimy jako ćwiczenia:

Lemat 18.4. *Jeżeli test Algorytmu nie powiedziecie z prawdopodobieństwem $\delta_0 < \frac{2}{9}$, to dla każdego x*

$$\mathbb{P}_{y,y'}[f(x+y) - f(y) \neq f(x+y') - f(y')] \leq 2\delta_0$$

Lemat 18.5. *Jeżeli dla każdego x*

$$\mathbb{P}_{y,y'}[f(x+y) - f(y) \neq f(x+y') - f(y')] < 4/9$$

to $\varphi(x)$ zdefiniowane jako najczęstsza wartość

$$\{f(x+y) - f(y) \ : \ y \in \mathbb{F}_2^n\}$$

spełnia

$$\mathbb{P}_y[\varphi(x) \neq f(x+y) - f(y)] < 1/3$$

Lemat 18.6. *Dla funkcji f, której test nie powiedzie się z prawdopodobieństwem $\delta_0 < 2/9$, funkcja φ zdefiniowana jako:*

$$\varphi(x) = \textit{najczęstsza wartość z } \{f(x+y) + f(y) \ : \ y \in \mathbb{F}_2^n\}$$

jest funkcją liniową.

18.2 Ogólniej: na przykładzie kodów Hadamarda

Podana poprzednio metoda działa, ale ciężko ją uogólnić na coś więcej (korzystamy z pomysłu podobnego do majority logice decoding).

Pokażemy teraz silniejsze twierdzenie, dowód będzie też ogólniejszy.

Twierdzenie 18.7. *Jeśli*

$$f : \mathbb{F}_2^n \to \mathbb{F}_2$$

spełnia

$$\mathbb{P}_{x,y}\left[f(x) + f(y) \neq f(x+y)\right] = \delta_0 > 0.$$

to

$$\delta(f, \mathcal{F}) \leq \delta_0 \ .$$

Zidentyfikujmy przeciwdziedzinę $\mathbb{F}_2$ z $\{\{-1,+1\},\cdot\}$. Od teraz będziemy myślimy o f jako o

$$f : \mathbb{F}_2^n \to \{-1,+1\} \subseteq \mathbb{R} \ .$$

Wtedy:

$$\mathbb{P}_{x,y}[f(x)\cdot f(y) = f(x+y)] = \mathbb{P}_{x,y}[f(x)\cdot f(y)\cdot f(x+y) = 1] \ .$$

Zauważmy, że jeśli mamy zmienną losową Z o wartościach $\{-1,1\}$, to

$$\begin{aligned}\mathbb{E}[Z] &= \mathbb{P}[Z=1] + (1 - \mathbb{P}[Z=1])\cdot(-1) \\ &= 1 + 2\mathbb{P}[Z=1] \ .\end{aligned}$$

I w takim razie

$$\mathbb{P}[Z=1] = \frac{1+\mathbb{E}[Z]}{2} \ .$$

W szczególności

$$\mathbb{P}_{x,y}[f(x)\cdot f(y)\cdot f(x+y)=1]=\frac{1+\mathbb{E}_{x,y}[f(x)\cdot f(y)\cdot f(x+y)]}{2}\ .$$

Zdefiniujmy iloczyn skalarny w tej przestrzeni funkcji

$$\langle f,g\rangle=\mathbb{E}_{x\in\mathbb{F}_2^n}[f(x)g(x)]\ .$$

Jeśli myślimy, że funkcja zapisana jest jak wektor wartości w punktach, to to jest zwykły iloczyn skalarny (suma iloczynów po współrzędnych), tylko przeskalowany. W szczególności faktycznie jest iloczynem skalarnym.

Zauważmy, że

Fakt 18.8. *Jeżeli*

$$f,g:\mathbb{F}_2^n\to\{-1,1\}\ \textit{to}\ \langle f,g\rangle=1-2\delta(f,g)\ .$$

Dowód pozostawiamy jako ćwiczenie.

Wiemy, że wszystkie homomorfizmy liniowe $f:\mathbb{F}_2^n\to\mathbb{F}_2$ są postaci $f=f_\alpha$, gdzie $f_\alpha(x)=\langle x,\alpha\rangle$ (czyli standardowy iloczyn skalarny). Po przepisaniu przeciwdziedziny na zbiór $\{-1,1\}$ dostajemy, że wszystkie odpowiadające funkcje (czyli homomorfizmy) są postaci

$$L(x)=L_\alpha(x)=(-1)^{\langle\alpha,x\rangle},$$

gdzie $\alpha=(\alpha_1,\ldots,\alpha_n)\in\mathbb{F}_2^n$. W szczególności

$$\delta(f,\mathcal{F})=\min_\alpha\delta(f,L_\alpha)$$

W zasadzie nie będziemy korzystali z definicji L_α po udowodnieniu, że funkcje te tworzą ortonormalna bazę funkcji $\mathbb{F}_2^n\to\mathbb{R}$.

Lemat 18.9. *Funkcje $\{L_\alpha\}$ stanowią bazę ortonormalną przestrzeni funkcji $\mathbb{F}_2^n\mapsto\mathbb{R}$ (z powyższym iloczynem skalarnym).*

Dowód pozostawiamy jako ćwiczenie.

Wprost z definicji dostajemy, że zachodzi

Fakt 18.10.

$$L_\alpha(x+y)=L_\alpha(x)L_\alpha(y)\ ,$$
$$L_{\alpha+\beta}(x+y)=L_\alpha(x)L_\beta(x)\ .$$

Oznaczmy też (współczynniki Fouriera):

$$\widehat{f}_\alpha=\langle f,\alpha\rangle=1-2\delta(f,L_\alpha)\ . \tag{18.1}$$

Wtedy:

$$f=\sum_\alpha\widehat{f}_\alpha L_\alpha\ .$$

Zauważmy, że wiele z tych współczynników może być ujemnych. Skorzystamy z następującego faktu (dowód jako ćwiczenie)

Lemat 18.11. *Dla każdej funkcji $f:\mathbb{F}_2^n\to\{-1,1\}$ istnieje L_α, takie że*

$$\delta(f,L_\alpha)\le\frac{1}{2}\ .$$

Wniosek 18.12. Dla każdej funkcji $f : \mathbb{F}_2^n \to \{-1, 1\}$ istnieje L_α, takie że

$$\widehat{f}_\alpha \geq 0 \ .$$

Skoro to jest baza, to trzeba sprawdzić, jak nasza funkcja zachowuje się na funkcjach bazowych:

$$\begin{aligned}
\mathbb{E}_{x,y}[L_\alpha(x)L_\beta(y)L_\gamma(x+y)] &= \mathbb{E}_{x,y}[L_\alpha(x)L_\beta(y)L_\gamma(x)L_\gamma(y)] \\
&= \mathbb{E}_x[L_\alpha(x)L_\gamma(x)] \cdot \mathbb{E}_y[L_\beta(y)L_\gamma(y)] \\
&= \langle L_\alpha, L_\gamma \rangle \langle L_\beta, L_\gamma \rangle \\
&= \begin{cases} 1 & \text{jeśli } \alpha = \gamma = \beta \\ 0 & \text{w p.p.} \end{cases}
\end{aligned}$$

I w takim razie

$$\begin{aligned}
\mathbb{E}_{x,y}[f(x)f(y)f(x+y)] &= \mathbb{E}_{x,y}\left[\sum_\alpha \widehat{f}_\alpha L_\alpha(x) \sum_\beta \widehat{f}_\beta L_\beta(y) \sum_\gamma \widehat{f}_\gamma L_\gamma(x+y)\right] \\
&= \sum_{\alpha,\beta,\gamma} \widehat{f}_\alpha \widehat{f}_\beta \widehat{f}_\gamma \mathbb{E}_{x,y}[L_\alpha(x)L_\beta(y)L_\gamma(x)L_\gamma(y)] \\
&= \sum_\alpha \widehat{f}_\alpha^3
\end{aligned}$$

W takim razie z Lematu i (18.1) mamy
Wracając do reprezentacji funkcji, z równości Parsevala mamy

$$\sum_\alpha \widehat{f}_\alpha^2 = \langle f, f \rangle = 1$$

i w takim razie

$$\begin{aligned}
\sum_\alpha \widehat{f}_\alpha^3 &\leq \sum_\beta \widehat{f}_\beta^2 (\max_\alpha \widehat{f}_\alpha) \\
&= \max_\alpha \widehat{f}_\alpha
\end{aligned}$$

Ale jednocześnie

$$\widehat{f}_\alpha = \langle f, L_\alpha \rangle = 1 - 2\delta(f, L_\alpha)$$

Czyli

$$\begin{aligned}
\max_\alpha \widehat{f}_\alpha &= 1 - 2 \min_\alpha \delta(f, L_\alpha) \\
&= 1 - 2\delta(f, \mathcal{F})
\end{aligned}$$

Czyli

$$\mathbb{E}_{x,y}[f(x)f(y)f(x+y)] \leq 1 - 2\delta(f, \mathcal{F})$$

Przepisując na prawdopodobieństwo

$$\begin{aligned}
2\mathbb{P}_{x,y}[f(x) \cdot f(y) \cdot f(x+y) = 1] - 1 &\leq 1 - 2\delta(f, \mathcal{F}) \\
\delta(f, \mathcal{F}) &\leq 1 - \mathbb{P}_{x,y}[f(x) \cdot f(y) \cdot f(x+y) = 1] \\
&= \mathbb{P}_{x,y}[f(x) \cdot f(y) \neq f(x+y)] \\
&= \delta_0 \ .
\end{aligned}$$

Rozdział 19

Sketching i kody korygujące błędy wstawiania

Na podstawie [2, 1].

19.1 Sketching

Problem jest następujące: chcemy zsynchronizować dwa pliki „w ustaloną stronę", tzn. serwer ma aktualną wersję pliku, zaś klient starą. Wiemy, że wykonano najwyżej k edycji, przy czym dopuszczamy następujące operacje:

insert wstawienie pojedynczej litery

edit usunięcie pojedynczej litery

edit zmiana pojedynczej litery (tym się w zasadzie nie będziemy zajmować, bo da się zrobić przy użyciu dwóch powyższych)

Definicja 19.1 (Odległość edycyjna)**.** Odległość edycji pomiędzy dwoma słowami s, t to minimalna liczba wstawień i usunięć potrzebnych do przekształcenia jednego w drugie. Oznaczenie: $ED(s, t)$.

Chcemy zsynchronizować te dwa pliki. Głównym kosztem będzie dla nas komunikacja, tj. w zasadzie nie interesuje nas, jak szybko to działa (choć jest wielomianowo).

Będziemy korzystać z funkcji haszujących; to da się ominąć, pewnym kosztem komunikacji i czasu, ale to temat na inny wykład. Zakładamy, że znamy funkcję (tj. obie strony znają), która dla dowolnego podciągu zwraca jego hasz wielkości $\mathcal{O}(\log n)$ bitów, tj.

$$f : \Sigma^* \to \{0,1\}^{c \log n}$$

taka że dla danych na wejściu słów w_s, w_k i ich dowolnych podsłów u_s, u_s zachodzi

$$f(u_s) \neq f(u_k)$$

(tak naprawdę potrzebujemy tylko dla słów tej samej długości, która jest potęgą dwójki).

Bez zmniejszenia ogólności zakładamy, że dane na wejściu słowa są tej samej długości n, która jest potęgą dwójki, oraz k też jest potęgą dwójki.

Korzystamy istotnie z następującej obserwacji:

Lemat 19.2. *Jeśli $ED(w_s, w_k) \leq k$ i podzielimy w_s na ℓ rozłącznych bloków, to w_k można przedstawić jako konkatenację $\leq \ell$ bloków, z których przynajmniej $\ell - k$ jest takich samych (i w tej samej kolejności), co w w_s, zaś pozostałe k można uzyskać z pozostałych k bloków w_s przy użyciu najwyżej k operacji edycji.*

Standardowy dowód pozostawimy jako ćwiczenie.

Uwaga: pozostałe bloki mogą być innej długości!

Protokół zaczyna od rozmiaru bloku $b = b_{\max} = n/4k$ i w każdej kolejnej fazie dzieli rozmiar bloku na dwa, aż do zejścia do $b_{\min}$, które jest rozmiaru obrazu funkcji haszującej (tzn. funkcja na nim już nic nie skraca). W każdej fazie połowimy b.

Protokół przedstawiamy w konwencji, w której w każdej fazie serwer wysyła wiadomość a klient coś liczy, ale to tylko opis: nie ma żadnej komunikacji zwrotnej, tak więc możemy przesłać to wszystko naraz.

W pierwszej rundzie dzielimy tekst na k bloków i wysyłamy hasze ich wszystkich. Klient liczy hasze wszystkich bloków długości b/k (uwzględniając różne możliwe przesunięcia!) identyfikuje te, które się zgadzają i „zapisuje" jako właściwe. Z założenia o funkcji haszującej, zidentyfikował dobrze. Klient zostawia sobie tylko niepoprawnie zidentyfikowane bloki.

Zgodnie z Lematem 19.2, klient ma najwyżej k takich bloków (i można je uzyskać z k bloków serwera poprzez k operacji edycji)

W każdej kolejne fazie serwer dzieli każdy blok na dwie równe części, haszuje każdy z nich i następnie koduje kolejne hasze systematycznym kodem RS o odległości $2k + 1$ (lub większej, ale po co) i wysyła tylko cyfry kontrolne, których jest $2k + 2$. Klient dzieli każdy swój nieznany blok na 2 części i ustala ich hasze jako „?" i korzystając z $2k + 2$ nowych haszy odtwarza te poprawne brakujące. Mając te hasze, znowu dopasowuje fragmenty.

Kończymy, gdy wielkość bloku wynosi k lub gdy hasz będzie tej wielkości, co blok.

Opisany powyżej protokół może być ulepszony, poprzez redukcję rozmiarów haszy do stałej niezależnej od n. Będzie to możliwe poprzez modyfikację protokołu przy użyciu następujących pomysłów, by uodpornić go na kolizje haszy:

- Przy mniejszych rozmiarach haszyszu, kolizje są nieuniknione. Jednakże, nawet jeśli klient nieprawidłowo odtwarza podciąg, wartości haszy odpowiadające jego podciągom na następnym poziomie nie zgodzi się z komunikatem serwera. W rzeczywistości, można wykazać, że jeśli serwer używa systematycznego kodu korekcji błędów, a nie tylko kodu korekcji błedów wymazywań, to klient będzie mógł wykryć i skorygować nieprawidłowo odtworzone wartości powstałych w wyniku użycia haszy kolizje na wcześniejszych poziomach.

- Używamy też tego, że dobrze odtworzone podciągi są w tej samej kolejności u klienta i u serwera.

Protokół może być nawet zderandomizowany.

19.2 Kody wstawiania

19.2.1 Na podstawie sketchingu.

Powyższy pomysł na sketching można przenieść też na kody korygujące błędy wstawiania oraz usuwania: przesyłamy najpierw wiadomość a następnie komunikację; z grubsza, jeśli błędy były w wiadomości, to można odtworzyć z komunikacji. Jeśli w samej komunikacji, to trzeba trochę popracować.

19.2.2 Indeksowanie

Tej odległości można użyć do zdefiniowania kodów korekcyjnych dla błędów usunięć i wstawień, analogicznie jak poprzednio.

Ogólnie próbuje się obecnie do nich podejść poprzez redukcję do klasycznych kodów korygujących, używając dodatkowych technik i pomysłów.

Chcemy przesłać ciąg symboli $m_1, \ldots, m_n$. Wyobraźmy sobie, że możemy zwiększyć alfabet, tak że może on zwierać symbol i numer pozycji, tj. przekazujemy $(m_1, 1), (m_2, 2), \ldots$ Odzyskujemy symbole w naturalny sposób, przy czym jeśli nie mamy nic dla indeksu i lub mamy dwa komunikaty, to uznajemy, że mamy wymazanie. (Zauważmy, że możemy mieć błąd w klasycznym sensie, jeśli usuniemy symbol dla pozycji i i dodamy nowy symbol na pozycji i). Będziemy mówić o półbłędach (dla klasycznych kodów korekcyjnych), tzn. wymazanie traktujemy jako pół błędu, a zamianę jak cały

Twierdzenie 19.3. *W powyższym schemacie indeksowania, jeśli będzie k błędów wstawień/usunięć, to podany algorytm wygeneruje komunikat z nie wiecej niż k półbłędami (w sensie zwykłych kodów korekcyjnych).*

Dowód. Każde wymazanie w zrekonstruowanym ciągu wymaga co najmniej jednego błędu na odpowiedniej pozycji. Każda zamiana to przynajmniej dwa błędy (wymazanie i wstawienie). □

Głównym problemem tego podejscia (poza niejasnym rozmiarem alfabetu) jest (liniowy) wzrost alfabetu wraz z n. Poprawimy to w następującej definicji.

19.2.3 Ciągi synchronizujące

Rozważmy ciąg S długości n oraz skojarzenie pomiędzy dwoma kopiami S, takie że symbole na końcach każdej krawędzi są identyczne i krawędzie nie przecinają się. Nazywamy każdy takie skojarzenie *samodopasowaniem* S. Każda krawędź w takim skojarzeniu, która łączy rózne pozycje, nazywamy *złą*

Definicja 19.4 (Ciąg ϵ-samo-podobny)**.** Ciąg S jest ϵ-samopodobny, jeżeli dla każdego samodopasowania liczba złych krawędzi jest nie większa niż ϵn.

Używając standardowych technik probabilistycznych można pokazać, że losowy ciąg długości n nad alfabetem q-elementowym z dużym prawdopodobieństwem jest $\Theta\left(\frac{1}{\sqrt{q}}\right)$-samopodobny.

Kodowanie jest naturalne: używamy ciągów synchronizujących do indeksowania. Aby poprawić otrzymana wiadomość, postępujemy w następujący sposób:

Algorytm 14

Założenie: Używamy ciągu S do synchronizacji; odczytaliśmy $(m_i, \widetilde{S}_i)_i$
for $m \leftarrow 1 \,..\, \frac{1}{\sqrt{\epsilon}}$ **do**
obliczy najdłuższe niekrzyżujące się skojarzenie pomiędzy S i $\widetilde{S}$
dla sparowanych $S_j, \widetilde{S}_i$ z wiadomością (m_i) przypisz m_i na pozycję j
usuń sparowane pozycje z $S, \widetilde{S}$

Twierdzenie 19.5. *Powyższy algorytm dla ciągu ϵ-samopodobnego i otrzymanego komunikatu, w którym nastąpiło k błędów usunięć i wstawień, zwraca ciąg w którym jest najwyżej $k + 4n\sqrt{\epsilon}$ pół błędów.*

Dowód. Zauważmy najpierw, że tak samo jak poprzednio, najwyżej k półbłędów pochodzi od samych błędów wstawień i usunięć, chcemy pokazać, że nasz sposób odtwarzania wprowadza najwyżej $4n\sqrt{\epsilon}$ pół błędów.

Są trzy sposoby otrzymania błędów wskutek nieprawidłowego dekodowania otrzymanych symboli:

1. Symbol (S_i, m_i) nie został zmieniony (i dotarł jako $(\widetilde{S}_j, m_i)$), ale $\widetilde{S}_j$ zostaje nieprawidłowo dopasowane do innego $S_{i'}$.

2. Symbol (S_i, m_i) nie został zmieniony (i dotarł jako $(\widetilde{S}_j, m_i)$), $\widetilde{S}_j$ nie zostało dopasowane ale S_i zostało dopasowane do czegoś innego.

3. Symbol (S_i, m_i) nie został zmieniony (i dotarł jako $(\widetilde{S}_j, m_i)$) ale nie zostanie skojarzone z żadną pozycję w czasie wszystkich rund algorytmu.

W pierwszym przypadku takie $(S_i, S_{i_0}$ jest złą krawędzią. Ponieważ S jest ϵ-samopodobny, to ma $\leq n\epsilon$ takich krawędzi. Rund jest w sumie $\frac{1}{\sqrt{\epsilon}}$ rund, łączna liczba takich błędnych dekodowań jest nie większa niż $\epsilon n \cdot \frac{1}{\sqrt{\epsilon}} = n\sqrt{\epsilon}$. Każdy liczy się jako dwa półbłędy.

W drugim przypadku analiza jak poprzednio, ale teraz to są wymazania, więc liczą się jako jeden półbłąd, tj. mamy $n\sqrt{\epsilon}$ półbłędów.

W drugim przypadku, jeśli na końcu procedury znajdują się ℓ takich symboli, to istnieje skojarzenie między ich źródłem a nimi, rozmiaru ℓ. Czyli każda poprzednia iteracja miała skojarzenie rozmiaru $\geq \ell$, czyli w szczególności $\frac{1}{\sqrt{\epsilon}}\ell \leq n \implies \ell \leq n\epsilon$. To są błędy wymazywania, czyli liczymy je jako $n\sqrt{\epsilon}$ półbłędów. □

Bibliografia

[1] Bernhard Haeupler. Introduction to coding theory, coding for insertions and deletions. URL: www.cs.cmu.edu/~venkatg/teaching/au18-coding-theory/lec-scribes/insdel-coding.pdf.

[2] Utku Irmak, Svilen Mihaylov, and Torsten Suel. Improved single-round protocols for remote file synchronization. In *INFOCOM*, pages 1665–1676. IEEE, 2005. URL: https://doi.org/10.1109/INFCOM.2005.1498448, doi:10.1109/INFCOM.2005.1498448.

[3] Antoni Kościelski. Kodowanie. notatki do wykładu, 2000.

[4] San Ling and Chaoping Xing. *Coding Theory: A First Course*. Cambridge University Press, 2004. doi:10.1017/CBO9780511755279.

[5] J. H. van Lint. *Introduction to Coding Theory*. Springer-Verlag, Berlin, Heidelberg, 2nd edition, 1991.

[6] Michael O'Sullivan. Coding theory. URL: https://mosullivan.sdsu.edu/Teaching/coding04.html.

[7] Madhu Sudan. Sub-linear time algebraic algorithms. notatki do wykładu, 2007. URL: http://phdopen.mimuw.edu.pl/index.php?page=z07w1#notes.

[8] Madhu Sudan Venkatesan Guruswami, Atri Rudra. *Essential Coding Theory*. 2019. URL: https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book/web-coding-book.pdf.

[9] Mary Wootters. Algebraic error correcting codes. 2019. URL: http://web.stanford.edu/~marykw/classes/CS250_W19/index.html.