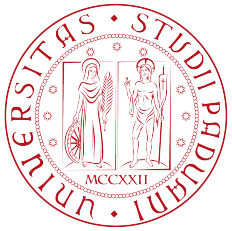

Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA 'TULLIO LEVI-CIVITA'
Corso di Laurea in Matematica

Tesi di Laurea

Un'introduzione alla teoria dei codici a correzione di errore

Candidato:
Alice Berti
Matricola 1201603

Relatore:
Prof.ssa
Eloisa Michela Detomi

Anno Accademico 2022-2023

Indice

Introduzione

In questa tesi mi propongo di presentare una breve introduzione alla teoria dei codici a correzione di errore. Lo scopo di un codice a correzione di errore è quello di identificare e correggere un certo numero di errori che si possono essere verificati nella trasmissione di un messaggio tramite un canale "*rumoroso*". Ne comprendiamo quindi l'importanza in svariati campi, dalle telecomunicazioni alla teoria dell'informazione, dall'informatica alla matematica stessa.

L'idea che sta alla base di un codice a correzione di errore è quella di trasmettere un messaggio in modo "*ridondante*", cioè aggiungendo un certo numero di cifre (se pensiamo ad esempio alla codifica binaria dei computer), che non contengono di per sè l'informazione originale ma che servono a ricostruirla una volta arrivata a destinazione, trovando e correggendo un numero massimo di errori, che dipende dalle proprietà del codice stesso scelto per la codifica. Questo processo ci permette di ritrovare con una buona precisione il messaggio iniziale, senza doverne chiedere il rinvio. A volte infatti questa non è nemmeno una possibilità: la ritrasmissione di un messaggio da un satellite che orbita intorno ad Urano può creare un ritardo di cinque ore, senza considerare gli ulteriori costi.

Ci occuperemo di capire come un messaggio possa essere codificato, trasmesso e infine decodificato tramite un codice e cercheremo di capire quanti e quali tipi di errore possano essere identificati e/o corretti e come, in base al tipo di codice che via via considereremo. Per raggiungere questo obiettivo mi baserò sul capitolo 14 del libro *Modern algebra with applications (second edition)* [2]. Amplierò la trattazione avvalendomi delle *Notes on coding theory* [3].

Infine fornirò due esempi di codici a correzione di errore, il Codice Fiscale italiano e il codice ISBN-13, ma con l'idea di presentarli a degli studenti delle scuole superiori, presupponendo quindi una conoscenza limitata della materia. L'obiettivo principale di quest'ultima parte del mio elaborato è spiegare in modo semplice ed accessibile uno dei tanti modi in cui la matematica permea la nostra vita quotidiana, nella speranza di trasmettere ad altri almeno un po' della bellezza che ci vedo io.

Ringraziamenti

Voglio ringraziare infinitamente Sara (cucci), per avermi convinta a continuare questo percorso anche quando pensavo di non farcela, per avermi sostenuta e supportata in questi anni, esame dopo esame.

Sono grata ai miei amici per la pazienza che hanno avuto nell'ascoltare i miei sfoghi e per il sostegno che mi hanno dato per superare i momenti difficili.

Ringrazio anche mia madre, che mi ha permesso di portare a termine questa laurea, senza avere il dubbio che non facesse per me.

Voglio ringraziare anche le mie due topine pelose Kirari e Yumeko, senza le quali non sarei riuscita a gestire il mio problema di ansia.

Capitolo 1

I codici a correzione di errore

In questo primo capitolo ci occuperemo di spiegare che cosa sono i codici a correzione di errore e a cosa servono (sezione 1.1). In secondo luogo presenteremo dei primi esempi di codici a correzione di errore (sezione 1.2) che serviranno a chiarire alcuni concetti teorici. Infine nelle sezioni 1.3 e 1.4 vedremo due possibilità di rappresentazione di un codice di n cifre binarie ed i codici lineari.

1.1 Il problema della codifica

Con il crescente impiego della tecnologia, nasce anche il bisogno di trasmettere le informazioni il più velocemente ed *accuratamente* possibile, ad esempio via radio o da / a dispositivi di archiviazione digitale. In molti sistemi di comunicazione le informazioni vengono trasmesse da una trasmettitore ad un ricevitore tramite un canale di trasmissione, il quale può subire l'effetto di ciò che chiamiamo *rumore*, cioè un qualcosa che può alterare l'informazione. In alcune situazioni è possibile chiedere nuovamente l'invio dell'informazione, mentre in altri è addirittura impossibile, o troppo costoso ritrasmettere il messaggio originale. Si pensi al caso in cui un messaggio contenente un errore sia conservato in un dispositivo di archiviazione : potrebbero passare mesi prima che venga letto e a quel punto il messaggio originale potrebbe essere già andato perso. Nasce da qui l'esigenza di poter capire se l'informazione arrivata presenta errori e se questi possano essere corretti.

Molti computer e sistemi di comunicazione utilizzano un sistema binario per gestire le informazioni, che saranno dunque stringhe di numeri formate dai simboli 0 e 1. Per questo motivo durante la trattazione mi limiterò al caso dei **codici binari**, anche se quasi tutto ciò vedremo si può generalizzare al caso di codici i cui simboli siano in un campo finito.

Siano $n, k \in \mathbb{N}^*$, con $k < n$. Per trasmettere un messaggio di lunghezza arbitraria tramite un canale rumoroso, lo dividiamo prima in blocchi di k cifre e codifichiamo ogni blocco aggiungendogli $n-k$ cifre di controllo. Otteniamo così una **parola codificata** o **code word** di n cifre. L'insieme di tutte le possibili code words prende il nome di **codice**.

Definizione 1.1. Chiamiamo un $(\mathbf{n}, \mathbf{k})$ **- codice** un codice le cui parole codificate sono lunghe n cifre, di cui k rappresentano l'informazione da codificare e $n-k$ sono di controllo.

Le parole così codificate possono ora essere trasmesse tramite un canale in cui è presente del rumore e, una volta ricevute, possono essere processate in due modi. Il codice infatti può essere utilizzato per accertare la presenza di errori, controllando se la parola ricevuta sia o no una parola codificata. Se così non fosse, allora possiamo concludere che un errore sia avvenuto e richiedere che la parola venga ritrasmessa. Possiamo però utilizzare il codice anche per correggere l'errore, scegliendo in fase di decodifica la parola codificata che con più probabilità può aver prodotto la parola che abbiamo ricevuto.

Definizione 1.2. Codici che usano questo tipo di riconoscimento e correzione dell'errore prendono il nome di **codici a correzione di errore** o, in inglese, **Error-Correcting Codes** (ECC).

In un (n, k) - codice, il messaggio originale è lungo k cifre e ci sono quindi 2^k possibili code words. Il messaggio ricevuto ha invece n cifre, quindi 2^n sarà il numero di parole che possono essere ricevute, anche se solamente 2^k sono code words. Le $n - k$ cifre che aggiungiamo sono chiamate **cifre ridondanti** poichè esse non aggiungono informazioni, ma permettono una maggiore accuratezza nella trasmissione del messaggio.

Definizione 1.3. Il valore $R = k/n$ prende il nome di **code rate** o **information rate**.

1.2 Primi esempi di codici semplici

Un primo esempio di codice che può essere ultilizzato per trovare un errore (error-detecting code) è il **parity check digit** o controllo di parità nelle cifre. Consideriamo un numero scritto in base 2, a cui aggiungiamo come ultima cifra una "cifra di controllo", che sia la somma modulo 2 delle altre cifre. In questo modo la somma delle cifre di un numero e la sua cifra di controllo sarà sempre pari, a meno che non si sia verificato un errore durante la trasmissione.

Notiamo però che questo tipo di controllo ci permette di rilevare solo un numero dispari di errori e nessun numero pari di errori.

Tabella 1.1: $(3, 2)$ Codice controllo di parità

$Message$	$CodeWord$
00	000
01	101
10	110
11	011

Tabella 1.2: $(3, 1)$ Repeating Code

$Message$	$CodeWord$
0	000
1	111

Proseguiamo con un confronto tra due codici di lunghezza 3, il $(3, 2)$ - codice ed il $(3, 1)$ - codice.

Il $(3, 2)$ - codice aggiunge all'inizio un'unica cifra, quella del controllo di parità, ad un messaggio di lunghezza 2. Pertanto la parola ricevuta sarà una code word solo se il numero di 1 è pari. Invece il $(3, 1)$ - codice aggiunge, al messaggio lungo una cifra, due cifre che lo ripetono (Repeating Code).

Nelle tabelle 1.1 e 1.2 sono riportati i messaggi e le code words dei due codici, dove la prima cifra della seconda colonna della tabella 1.1 è la cifra di controllo di parità. Notiamo che se si verifica un errore nella trasmissione nel $(3, 2)$ - codice, ad esempio 101 viene cambiato in 100, questo viene subito trovato, poichè la parola ricevuta ha un numero dispari di 1. Ma non c'è modo di sapere quale fosse il messaggio originale, dato che 100 ha la stessa probabilità di provenire da 110, 000 e 101. Inoltre con questo codice non è possibile capire quando sono avvenuti due errori. Quindi una volta ricevuto il messaggio prima si effettua un controllo di parità : se il numero di 1 è pari, la parola viene decodificata come le ultime due cifre; se invece il numero di 1 è dispari si sa solo che è avvenuto un errore.

Il $(3, 1)$ - codice invece può trovare uno o due errori di trasmissione ma può correggerne solo uno. Infatti in fase di decodifica, se il numero di 1 è maggiore a quello degli 0 si assume che il messaggio fosse 1, altrimenti che fosse 0. Ciò porta però a decodificare in modo errato il messaggio nel caso gli errori fossero due.

Ci chiediamo quindi come scoprire quali siano le capacità di trovare e correggere gli errori di un certo codice. Definiamo quindi

Definizione 1.4. La **distanza di Hamming** tra due parole u e v della stessa lunghezza è il numero di cifre in cui esse differiscono e la denotiamo con $d(u, v)$.

Possiamo interpretare la distanza di Hamming tra due parole come il numero di errori necessari per cambiare una nell'altra tramite il cambio di una cifra per volta. Nel $(3, 2)$ - codice la distanza tra due parole è sempre 2 : ad esempio $d(101, 110) = 2$.

Iniziamo a vedere una prima proposizione che collega la distanza di Hamming tra le parole di un codice e la capacità di quest'ultimo di trovare gli errori.

Proposizione 1.5. *Un codice trova tutti i possibili set di t o meno errori se e solo se la distanza di Hamming minima tra le code words è almeno $t + 1$.*

Dimostrazione. Se si verificano r errori quando la code word u viene trasmessa, la parola ricevuta v si troverà ad una distanza di Hamming r da u. Questi errori verranno trovati se e solo se v non è una code word. Quindi t o meno errori nella parola u verranno trovati se e solo se la distanza di Hamming di u è almeno $t + 1$ da ogni altra code word. □

Proposizione 1.6. *Un codice può trovare e correggere tutti i set di t o meno errori se e solo se la distanza di Hamming minima tra le code words è almeno $2t + 1$.*

Dimostrazione. Supponiamo che il codice contenga due parole u_1, u_2 tali che $d(u_1, u_2) \leq 2t$. Allora esiste una parola ricevuta v che differisce da u_1, u_2 in t o meno posizioni, quindi può aver avuto origine con t o meno errori sia da u_1 che

da u_2. Pertanto non verrà decodificata in modo corretto in nessuna delle due situazioni.

Invece, ogni codice la cui distanza di Hamming minima tra due code words sia $2t+1$ è in grado di correggere fino a t errori, scegliendo ogni volta la code word che sia più vicina a quella ricevuta. □

1.3 La rappresentazione polinomiale

Risulta utile scrivere una parola di n cifre binarie come un polinomio di grado minore di n a coefficienti in $\mathbb{Z}_2$ campo con due elementi. La parola $a_0a_1\dots a_{n-1}$ può essere rappresentata dal polinomio

$$a_0 + a_1x + \cdots + a_{n-1}x^{n-1} \in \mathbb{Z}_2[x].$$

Utilizziamo questa rappresentazione per mostrare come costruire alcuni codici.

Sia $p(x) \in \mathbb{Z}_2[x]$ un polinomio di grado $n-k$. Il **codice polinomiale generato** da $p(x)$ è un (n,k) - codice le cui code words sono polinomi, di grado strettamente minore di n, che sono divisibili per $p(x)$.

Un messaggio di lunghezza k è rappresentato da un polinomio $m(x)$, di grado minore di k, $\deg m(x) < k$. Vogliamo che l'informazione sia contenuta nei coefficienti di grado massimo, pertanto moltiplichiamo $m(x)$ per x^{n-k}, cosicchè $\deg\left(m(x)x^{n-k}\right) < n$. Per codificare quindi il messaggio $m(x)$ effettuiamo la divisione tra $x^{n-k}m(x)$ e $p(x)$ ed aggiungiamo il resto $r(x)$ a $x^{n-k}m(x)$ per formare il codice polinomiale

$$v(x) = r(x) + x^{n-k}m(x).$$

Il codice polinomiale è sempre un multiplo di $p(x)$, poichè

$$x^{n-k}m(x) = q(x)p(x) + r(x) \text{ dove } deg(r(x)) < n-k \text{ oppure } r(x) = 0$$

Pertanto si ha, tenuto conto che $r(x) = -r(x) \in \mathbb{Z}_2[x]$,

$$v(x) = r(x) + x^{n-k}m(x) = -r(x) + x^{n-k}m(x) = q(x)p(x).$$

Inoltre il polinomio $x^{n-k}m(x)$ ha zeri negli $n-k$ termini di ordine minore, mentre il polinomio $r(x)$ è di grado minore di $n-k$. Pertanto i k coefficienti di ordine maggiore di $v(x)$ sono le cifre che rappresentano il messaggio, mentre gli $n-k$ coefficienti di ordine minore sono le cifre di controllo; più precisamente queste sono i coefficienti del resto $r(x)$.

Il polinomio generatore $p(x) = a_0 + a_1x\cdots + a_{n-k}x^{n-k}$ è scelto sempre di modo che $a_0 = 1$ e $a_{n-k} = 1$ per due motivi. Il primo è evitare di sprecare cifre di controllo; il secondo è che se $a_0 = 0$ o $a_{n-k} = 0$ ogni codice polinomiale sarebbe multiplo di x oppure il coefficiente di x^{n-k-1} sarebbe sempre nullo.

Un messaggio ricevuto $u(x)$ può quindi essere testato per la presenza di errori semplicemente effettuando la divisione per il polinomio generatore $p(x)$. Se il resto è un polinomio non nullo, deve esserci stato un errore durante la trasmissione; se invece il resto è 0 allora $u(x)$ è una code word e non è avvenuto nessun errore o c'è stato un errore che non è trovabile in questo modo.

Cerchiamo ora di capire come si possa scegliere un polinomio generatore $p(x)$ in modo che il codice abbia delle buone proprietà senza dover aggiungere troppe cifre di controllo.

Proposizione 1.7. *Il polinomio $p(x) = 1 + x$ genera il $(n, n-1)$ - codice del controllo di parità.*

Dimostrazione. In generale un polinomio $f(x) \in \mathbb{Z}_2[x]$ è divisibile per $x + 1$ se e solo se $f(1) = 0$ e ciò si verifica se e solo se f contiene un numero pari di coefficienti non nulli. Perciò le code words generate da $p(x) = x + 1$ sono quelle che contengono un numero pari di 1. La cifra di controllo per il messaggio polinomiale $m(x)$ è il resto della divisione di $xm(x)$ per $p(x) = 1 + x$. Visto che

$$xm(x) = p(x) \cdot q(x) + r(x) \text{ e } p(1) = 0$$

la cifra di controllo è $r(1) = 1 \cdot m(1) = m(1)$, cioè la parità del numero di 1 nel messaggio. Questo codice è quindi il codice del controllo di parità. □

Esempio. Il $(3, 1)$ - codice visto nella sezione 1.2 che ripete il messaggio di una singola cifra tre volte e che ha solo due code words (tabella 1.2) è generato dal polinomio $p(x) = 1 + x + x^2$.

Forniamo infine un metodo, che fa uso dei polinomi primitivi, per trovare un generatore per un codice che identifichi uno, due o tre errori. Il grado del polinomio generatore sarà il più piccolo possibile, di modo che le cifre di controllo siano ridotte al minimo. Useremo senza dimostrare il seguente fatto :

Osservazione. Un polinomio irriducibile $p(x) \in \mathbb{Z}_2[x]$ di grado m è primitivo se $p(x) \mid (1 + x^k)$ per $k = 2^m - 1$ e per nessun k minore di questo.

Teorema 1.8. *Se $p(x)$ è un polinomio primitivo di grado m , allora il $(n, n-m)$ - codice generato da $p(x)$ trova tutti gli errori singoli e doppi se $n \leq 2^m - 1$.*

Dimostrazione. Sia $v(x)$ la parola trasmessa e $u(x) = v(x)+e(x)$ quella ricevuta, in cui il polinomio $e(x)$ è chiamato **polinomio errore**. L'errore è trovabile se e solo se $p(x) \nmid u(x)$. Dato che $p(x) \mid v(x)$, l'errore è trovabile se e solo se $p(x) \nmid e(x)$.

Se si verifica un singolo errore, $e(x)$ conterrà un solo termine, sia esso x^i per $0 \leq i < n$. Il fatto che $p(x)$ sia irriducibile implica che 0 non sia una sua radice, pertanto $p(x) \nmid x^i$ e l'errore è identificabile.

Se si verificano due errori, $e(x)$ avrà la forma $x^i + x^j$ dove $0 \leq i < j < n$. Quindi $e(x) = x^i(1 + x^{j-i})$ con $0 < j - i < n$. Ora $p(x) \nmid x^i$ e dato che $p(x)$ è primitivo, $p(x) \nmid (1+x^{j-i})$ se $j-i < 2^m-1$. Visto che $p(x)$ è irriducibile, $p(x) \nmid x^i(1 + x^{j-i})$ per $n \leq 2^m - 1$ e ne segue che ogni doppio errore è trovabile. □

Corollario 1.9. *Se $p_1(x)$ è un polinomio primitivo di grado m, il $(n, n-m-1)$ - codice generato da $p(x) = (1+x)p_1(x)$ trova ogni errore doppio ed ogni numero dispari di errori per $n \leq 2^m - 1$.*

Dimostrazione. Le code words generate da $p(x)$ devono essere divisibili sia per $p_1(x)$ che per $1 + x$. Il fattore $1 + x$ ha come effetto quello di aggiungere un'ulteriore cifra per il controllo di parità al codice. Perciò tutte le code words hanno un numero pari di termini ed il codice identificherà ogni numero dispari di errori. Dato che le code words devono essere divisibili per il polinomio primitivo $p_1(x)$, il codice troverà ogni errore doppio per $n \leq 2^m - 1$ in forza del teorema precedente 1.8. □

1.4 La rappresentazione matriciale e i codici lineari

Un altro modo per rappresentare la parola $a_1 \dots a_n$ di lunghezza n è con un elemento $\begin{pmatrix} a_1 & a_2 & \dots & a_n \end{pmatrix}^T$ dello spazio vettoriale $\mathbb{Z}_2^n = \mathbb{Z}_2 \times \mathbb{Z}_2 \times \cdots \times \mathbb{Z}_2$ di dimensione n su $\mathbb{Z}_2$.

In un (n, k) - codice, i 2^k possibili messaggi di lunghezza k sono tutti i gli elementi dello spazio $\mathbb{Z}_2^k$, mentre le 2^n possibili parole ricevute di lunghezza n compongono lo spazio $\mathbb{Z}_2^n$. Una codifica è quindi una funzione iniettiva

$$\gamma \colon \mathbb{Z}_2^k \to \mathbb{Z}_2^n$$

che assegna ad ogni messaggio di k cifre una code word di n cifre.

Definizione 1.10. Un (n, k) - codice è detto **codice lineare** se la funzione che codifica è un'applicazione lineare da $\mathbb{Z}_2^k$ a $\mathbb{Z}_2^n$.

Proposizione 1.11. *Sia $p(x)$ un polinomio di grado $n-k$ che genera un (n, k) - codice. Allora questo codice è lineare.*

Dimostrazione. Sia $\gamma \colon \mathbb{Z}_2^k \to \mathbb{Z}_2^n$ la funzione codifica definita dal polinomio generatore $p(x)$. Siano $m_1(x)$ e $m_2(x)$ due polinomi che rappresentano due messaggi entrambi di grado minore di k e siano m_1 ed m_2 gli stessi messaggi ma considerati come vettori in $\mathbb{Z}_2^k$. Il vettore codidificato $\gamma(m_i), i = 1, 2$ corrisponde al polinomio codificato $v_i(x) = r_i(x) + x^{n-k} m_i(x)$, dove $r_i(x)$ è il resto della divisione tra $x^{n-k} m_i(x)$ e $p(x)$. Ora

$$v_1(x) + v_2(x) = r_1(x) + r_2(x) + x^{n-k}[m_1(x) + m_2(x)]$$

e $r_1(x) + r_2(x)$ ha grado minore di $n - k$. Quindi $r_1(x) + r_2(x)$ è il resto della divisione tra $x^{n-k} m_1(x) + x^{n-k} m_2(x)$ e $p(x)$. Ne segue che $v_1(x) + v_2(x)$ corrisponde al vettore codificato $\gamma(m_1 + m_2)$ e

$$\gamma(m_1 + m_2) = \gamma(m_1) + \gamma(m_2).$$

Dato che gli unici scalari nel campo sono 0 e 1, ciò implica che γ è un'applicazione lineare. □

Definizione 1.12. Sia $\{e_1, e_2, \dots, e_n\}$ la base canonica di Z_2^n e sia G la matrice $n \times k$ che rappresenta, in questa base, la trasformazione $\gamma \colon \mathbb{Z}_2^k \to \mathbb{Z}_2^n$, definita da un (n, k) - codice lineare. Chiamiamo G **generator matrix** o **encoding matrix** del codice in inglese, **matrice generatrice** in italiano.

Osservazione. Se m è il vettore che rappresenta il messaggio, la code word rispettiva è $v = Gm$; i vettori codificati sono i vettori dell'immagine di γ e formano un sottospazio vettoriale di $\mathbb{Z}_2^n$ di dimensione k. Le colonne di G sono una base per questo sottospazio : pertanto un vettore è un vettore codificato se e solo se è combinazione lineare delle colonne della matrice generatrice G.

Esempio. Nel (3.2) - codice del controllo di parità, un vettore $m = (m_1, m_2)^T$ è codificato come $v = (c, m_1, m_2)^T$ con $c = m_1 + m_2$ la cifra del controllo di parità. Quindi la matrice generatrice G è

$$G = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ infatti } \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \end{pmatrix} = \begin{pmatrix} c \\ m_1 \\ m_2 \end{pmatrix}$$

Da questo breve esempio notiamo che se la parola codificata contiene le cifre del messaggio nelle ultime k posizioni, la matrice G avrà necessariamente la forma $G = \begin{pmatrix} P \\ \mathbb{I}_k \end{pmatrix}$, con P matrice $(n-k) \times k$ e $\mathbb{I}_k$ matrice identità $k \times k$.

Risulta quindi molto facile, dato un qualunque messaggio m, codificarlo : basta infatti calcolare Gm. Risulta più complesso invece, dato un vettore u, determinare dalla matrice G se u sia o meno un vettore codificato (o code vector). Abbiamo visto che questi formano un sottospazio di $\mathbb{Z}_2^n$, $Im\gamma$, di dimensione k, generato dalle colonne di G.

Cerchiamo quindi un'applicazione lineare $\eta \colon \mathbb{Z}_2^n \to \mathbb{Z}_2^{n-k}$, rappresentata da una matrice H, il cui kernel sia esattamente $Im\gamma$. Così facendo un vettore u è un code vector se e solo se $Hu = 0$. Questa scelta di H motiva anche il punto *2.* del seguente teorema.

Teorema 1.13. *Sia $\gamma \colon \mathbb{Z}_2^k \to \mathbb{Z}_2^n$ la funzione codificatrice di un (n,k) - codice lineare, con matrice generatrice $G = \begin{pmatrix} P \\ \mathbb{I}_k \end{pmatrix}$, con P matrice $(n-k) \times k$ e $\mathbb{I}_k$ matrice identità $k \times k$. L'applicazione lineare $\eta \colon \mathbb{Z}_2^n \to \mathbb{Z}_2^{n-k}$ definita dalla matrice $H = \begin{pmatrix} \mathbb{I}_{n-k} & P \end{pmatrix} \in M_{(n-k)\times n}(\mathbb{Z}_2)$ ha le seguenti proprietà:*

1. *$Ker\eta = Im\gamma$.*
2. *Un vettore ricevuto u è un code vector se e solo se $Hu = 0$.*

Dimostrazione. La funzione composta $\eta \circ \gamma \colon \mathbb{Z}_2^k \to \mathbb{Z}_2^{n-k}$ è identicamente nulla poichè, utilizzando la moltiplicazione a blocchi per le matrici a coefficienti nel campo $\mathbb{Z}_2$ si ottiene :

$$HG = \begin{pmatrix} \mathbb{I}_{n-k} & P \end{pmatrix} \begin{pmatrix} P \\ \mathbb{I}_k \end{pmatrix} = (\mathbb{I}_{n-k}P + P\mathbb{I}_k) = P + P = 0$$

Ne segue che $Im\gamma \subseteq Ker\eta$. Le prime $n-k$ colonne di H sono esattamente la base canonica di $\mathbb{Z}_2^{n-k}$, $Im\eta$ genera quindi $\mathbb{Z}_2^{n-k}$ e ha dimensione $\dim Im\eta = n-k$. Per il teorema del rango, che dice che $\dim \mathbb{Z}_2^n = \dim Ker\eta + \dim Im\eta$, si ha:

$$\dim Ker\eta = \dim \mathbb{Z}_2^n - \dim Im\eta = n - (n-k) = k.$$

Ma anche $\dim Im\gamma = k$ e sono uno contenuto nell'altro, quindi sono uguali. □

Definizione 1.14. La matrice H di dimensione $(n-k) \times n$ definita nel teorema precedente 1.13 si chiama **matrice del controllo di parità** (o **parity check matrix**) del (n,k) - codice.

Esempio. Riprendiamo l'esempio del $(3,2)$ - codice. In questo caso la matrice del controllo di parità è la matrice 3×1 $H = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$. Un vettore ricevuto $u = (u_1, u_2, u_3)^T$ è un code vector se e solo se

$$Hu = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = u_1 + u_2 + u_3 = 0.$$

Possiamo riassumere quanto visto finora in questa sezione 1.4 così : se $G = \begin{pmatrix} P \\ \mathbb{I}_k \end{pmatrix}$ è la matrice generatrice di un (n,k) - codice, allora $H = \begin{pmatrix} \mathbb{I}_{n-k} & P \end{pmatrix}$ è

la matrice di controllo di parità. Codifichiamo un messaggio m calcolando Gm e troviamo errori in un vettore ricevuto u calcolando Hu. Un codice lineare è determinato dall'assegnazione di G o di H.

Aggiungiamo un breve risultato riguardante i codici lineari, noto come SINGLETON BOUND, che fornisce una limitazione dall'alto della distanza di Hamming minima tra le parole codificate.

Teorema 1.15. *(SINGLETON BOUND.) Sia dato un (n, k) - codice lineare su un campo F. Allora la minima distanza di Hamming (d_{min}) tra due code words soddisfa*

$$d_{min} \leq n - k + 1.$$

Dimostrazione. Ogni sottomatrice $(n-k) \times (n-k+1)$ della matrice di controllo H (che può essere definita su un campo generico $F \neq \mathbb{Z}_2$ e che in questo caso non chiamiamo "di parità") ha rango al massimo $n-k$. Ogni insieme di $n-k+1$ colonne della matrice di controllo è quindi linearmente dipendente.

Ora nel Lemma 2.1 vedremo che d_{min} tra due code words di un codice lineare è il minimo numero di entrate diverse da 0 nelle code words non nulle. Poichè H ha $n-k+1$ colonne linearmente dipendenti allora esiste un vettore u per cui $Hu = 0$, che ci fornisce una combinazione lineare di colonne di H con coefficienti u_i dei quali esattamente $n-k+1$ sono non nulli. Per quanto visto nel Teorema 1.13, u è una code word e $d_{min} \leq n - k + 1$. □

Correzione degli errori e decodifica

Vorremmo trovare un metodo efficiente per correggere gli errori e decodificare le informazioni. Un primo modo, un po' "brutale", sarebbe calcolare la distanza di Hamming tra la parola ricevuta ed ogni code word; si assume poi che quella più vicina alla parola ricevuta sia la parola trasmessa. Il problema diventa però enorme a livello computazionale non appena la lunghezza del messaggio cresce abbastanza.

Consideriamo un (n, k) - codice con funzione codificatrice $\gamma \colon \mathbb{Z}_2^k \to \mathbb{Z}_2^n$. Sia $V = Im\gamma$ il sottospazio generato dai code vectors. Se il vettore $v \in V$ é inviato tramite il canale di comunicazione e l'errore $e \in \mathbb{Z}_2^n$ occorre durante la trasmissione, il vettore ricevuto sarà $u = v + e$. In fase di decodifica del vettore u bisogna capire quale sia il vettore v che con più probabilità è stato trasmesso e quale sia l'errore e che con più probabilità può essere successo. In particolare non sappiamo con certezza quale sia il vettore v, ma sappiamo che il vettore e è tale per cui $e = -v + u = v + u$, cioè $e \in V + u$, classe laterale di V in $\mathbb{Z}_2^n$.

Definizione 1.16. Definiamo il **coset leader** (o leader della classe laterale) come il pattern di errori che si verifica con più probabilità nella classe laterale di V in $\mathbb{Z}_2^n$. In particolare ogni classe laterale avrà un suo coset leader.

Il coset leader di solito è l'elemento con più entrate nulle. Se due elementi della stessa classe laterale hanno lo stesso numero di entrate diverse da 0, si può scegliere quale dei due considerare il coset leader arbitrariamente.

Le classi laterali di V in $\mathbb{Z}_2^n$ possono essere caratterizzate in termini della matrice del controllo di parità H. Il sottospazio V, come visto nel teorema 1.13, è il kernel dell'applicazione $\eta \colon \mathbb{Z}_2^n \to \mathbb{Z}_2^{n-k}$. Quindi, per il primo teorema di omomorfismo, l'insieme delle classi laterali $\mathbb{Z}_2^n/V$ è isomorfo a $Im\eta$, con

l'isomorfismo definito da $V + u \mapsto \eta(u) = Hu$. Perciò la classe laterale $V + u$ è caratterizzata dal vettore Hu.

Definizione 1.17. Se H è la matrice di controllo di parità $(n-k)\times n$ ed $u \in \mathbb{Z}_2^n$, allora il vetttore Hu è chiamato **sindrome di u**.

Ogni elemento di $\mathbb{Z}_2^{n-k}$ è una sindrome; quindi ci sono 2^{n-k} diverse classi laterali e 2^{n-k} diverse sindromi.

Teorema 1.18. *Due vettori sono nella stessa classe laterale di V in $\mathbb{Z}_2^n$ se e solo se hanno la stessa sindrome.*

Dimostrazione. Se $u_1, u_2 \in \mathbb{Z}_2^n$ allora sono equivalenti:

1. $V + u_1 = V + u_2$
2. $u_1 - u_2 \in V$
3. $H(u_1 - u_2) = 0$
4. $Hu_1 = Hu_2$

□

Possiamo quindi decodificare le parole ricevute utilizzando la seguente procedura:

1. Calcolare la sindrome della parola ricevuta.
2. Trovare il leader della classe laterale nella classe laterale corrispondente alla sindrome calcolata nel punto 1.
3. Sottrarre il coset leader alla parola ricevuta per ottenere la parola che più probabilmente è stata trasmessa.
4. Eliminare le cifre di controllo per ottenere il messaggio trasmesso con più probabilità.

Per un codice polinomiale generato da $p(x)$, la sindrome del polinomio ricevuto $u(x)$ è il resto $r(x)$, rifacendoci alle notazioni utilizzate nella sezione 1.3. Infatti la j-esima colonna della matrice H è il resto della divisione di x^{j-1} per $p(x)$. Perciò la sindrome degli elementi di un codice polinomiale può essere calcolata facilmente.

Capitolo 2

Codici BCH

La classe di codici a correzione di errore più potente finora conosciuta (almeno fino al 2004, anno di pubblicazione di *Modern algebra with applications (second edition)*) fu scoperta nel 1960 dal matematico francese Alexis Hocquenghem ed indipendentemente dai matematici Raj Chandra Bose e D.K. Ray-Chaudhuri. Prendono il nome di **BCH codici**, dalle iniziali dei cognomi dei loro inventori.

Per ogni intero positivo m e t , con $t < 2^{m-1}$, esiste un BCH codice di lunghezza $n = 2^m - 1$ che corregge ogni combinazione di t o meno errori. Questi codici sono codici polinomiali con un generatore $p(x)$ di grado $\leq mt$ e che hanno messaggi di lunghezza almeno $n - mt$.

Un BCH codice a t-correzione di errore di lunghezza $n = 2^m - 1$ ha polinomio generatore $p(x)$ costruito come segue. Sia α un elemento primitivo del gruppo $Gal(\mathbb{F}_{2^m}/\mathbb{F}_2)$, cioè tale che $\mathbb{F}_{2^m}/\mathbb{F}_2 = \mathbb{F}_2(\alpha)$. Sia $p_i(x) \in \mathbb{Z}_2[x]$ il polinomio irriducibile che ha α^i come radice e definiamo

$$p(x) = mcm(p_1(x), p_2(x), \ldots, p_{2t}(x)).$$

É chiaro che $\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2t}$ sono radici di $p(x)$. Siccome vale $[p_i(x)]^2 = p_i(x^2)$, ne segue che α^{2i} è radice di $p_i(x)$. Quindi

$$p(x) = mcm(p_1(x), p_3(x), \ldots, p_{2t-1}(x)). \tag{2.1}$$

Dato che $Gal(\mathbb{F}_{2^m}/\mathbb{F}_2)$ è uno spazio vettoriale di dimensione m sul campo $\mathbb{F}_2(= \mathbb{Z}_2)$ per ogni $\beta = \alpha^i$ gli elementi $1, \beta, \beta^2, \ldots, \beta^m$ sono linearmente dipendenti. L'elemento β soddisfa un polinomio di grado al massimo m in $\mathbb{Z}_2[x]$ e i polinomi irriducibili $p_i(x)$ devono avere grado al massimo m. Perciò

$$deg\, p(x) \leq deg\, p_1(x) \cdot deg\, p_3(x) \cdot \ldots \cdot deg\, p_{2t-1}(x) \leq mt.$$

Dimostriamo ora che il BCH codice generato da un tale $p(x)$ corregge effettivamente t errori.

Lemma 2.1. *La minima distanza di Hamming tra due code word di un codice lineare è il minimo numero di entrate diverse da* 0 *nelle code word non nulle.*

Dimostrazione. Se v_1 e v_2 sono code words, dato che il codice è lineare, anche $v_1 - v_2$ è una code word. La distanza di Hamming tra v_1 e v_2 è il numero di 1 in $v_1 - v_2$. Ne segue la tesi poichè la parola nulla (quella che ha tutte entrate

0) è sempre una code word, la cui distanza di Hamming da un'altra parola è il numero di 1 di quest'ultima. □

Teorema 2.2. *Se $t < 2^{m-1}$, la minima distanza tra code words in un BCH codice generato dal polinomio nella formula 2.1 è almeno $2t+1$. Perciò il codice corregge t o meno errori.*

Dimostrazione. Supponiamo che il codice contenga un polinomio codificato con meno di $2t+1$ termini non nulli e sia questo

$$v(x) = v_1 x^{r_1} + \cdots + v_{2t} x^{r_{2t}} \text{ dove } r_1 < \cdots < r_{2t}.$$

Questo polinomio è divisibile per il generatore $p(x)$, ha quindi come radici $\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2t}$. Se $1 \leq i \leq 2t$,

$$\begin{aligned} v(\alpha^i) &= v_1 \alpha^{ir_1} + \cdots + v_{2t} \alpha^{ir_{2t}} \\ &= \alpha^{ir_1}(v_1 + \cdots + v_{2t} \alpha^{ir_{2t} - ir_1}). \end{aligned}$$

Chiamiamo $s_i = r_i - r_1$ e notiamo che gli elementi $v_1, \ldots, v_{2t}$ soddisfano le seguenti equazioni lineari :

$$\begin{cases} v_1 + v_2 \alpha^{s_2} + \cdots + v_{2t} \alpha^{s_{2t}} & = 0 \\ v_1 + v_2 \alpha^{2s_2} + \cdots + v_{2t} \alpha^{2s_{2t}} & = 0 \\ \vdots \quad \vdots & \vdots \\ v_1 + v_2 \alpha^{2ts_2} + \cdots + v_{2t} \alpha^{2ts_{2t}} & = 0 \end{cases} \tag{2.2}$$

La matrice dei coefficienti del sistema è non singolare poichè il suo determinante è quello di una matrice di Vandermonde:

$$det \begin{pmatrix} 1 & \alpha^{s_2} & \ldots & \alpha^{s_{2t}} \\ 1 & \alpha^{2s_2} & \ldots & \alpha^{2s_{2t}} \\ \vdots & & & \vdots \\ 1 & \alpha^{2ts_2} & \ldots & \alpha^{2ts_{2t}} \end{pmatrix} = \prod_{2 \leq j < i \leq 2t} (\alpha^{s_i} - \alpha^{s_j}) \neq 0.$$

Il determinante è non nullo perchè $\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2t}$ sono tutte distinte se $t < 2^{m-1}$. Quindi il sistema lineare 2.2 ha unica soluzione $v_1 = v_2 = \cdots = v_{2t} = 0$. Concludiamo che non esistono code words non identicamente nulle con meno di $2t+1$ termini non nulli. In forza del Lemma 2.1 e della Proposizione 1.6 deduciamo che il codice corregge effettivamente t o meno errori. □

Concludiamo questo capitolo con un breve ma significativo esempio.

Esempio. Un BCH $(127, 92)$- codice aggiunge 35 cifre di controllo alle 92 del messaggio e corregge fino a 5 errori. Riprendendo quanto visto nella sezione 1.4, ciò implica l'esistenza di 2^{35} possibili sindromi. Diventa quasi impossibile memorizzare tutte le sindromi e i corrispettivi coset leaders in un computer e l'operazione di decodifica dovrà essere effettuata in altro modo. Gli errori, con i BCH codici, possono essere trovati con mezzi algebrici senza che sia necessario elencare sindromi e coset leaders uno ad uno. In generale, un codice con un information rate relativamente alto necessariamente è lungo; di conseguenza, perchè il codice sia utile, deve possedere un algoritmo algebrico di decodifica più semplice possibile.

Capitolo 3

Codici Reed-Solomon

Nel 1960 la coppia di matematici statunitensi I. S. Reed e G. Solomon, che al tempo lavoravano presso il *Lincoln Laboratory* del Massachussets Institute of Technology (MIT), introducono una famiglia di codici a correzione di errore che si rivelano molto utili per le applicazioni pratiche.

Nella prima sezione 3.1 ci proponiamo di descrivere le proprietà di base e la struttura dei cosiddetti ***codici Reed-Solomon generalizzati***. Nella sezione seguente 3.2 invece affronteremo un metodo algebrico di decodifica particolarmente efficiente.

3.1 Struttura e proprietà dei codici Reed-Solomon Generalizzati

Sia F un campo, $v_1, \ldots, v_n \in F$ scalari non nulli e $\alpha_1, \ldots, \alpha_n \in F$ scalari tutti distinti. Denotiamo poi $v = (v_1, \ldots, v_n)$ e $\alpha = (\alpha_1, \ldots, \alpha_n)$.

Definizione 3.1. Per $0 \leq k \leq n$ definiamo i **codici di Reed-Solomon Generalizzati**, o in breve **GRS**, come segue

$$GRS_{n,k}(\alpha, v) = \{\big(v_1 f(\alpha_1), v_2 f(\alpha_2), \ldots, v_n f(\alpha_n)\big) \in F^n \mid f(x) \in F[x]_k\},$$

dove con la scrittura $F[x]_k$ intendiamo i polinomi in $F[x]$ di grado minore o uguale a k, spazio vettoriale di dimensione k sul campo F.

Per n, α, v fissati, i vari codici GRS godono della proprietà $GRS_{n,k-1}(\alpha, v) \subseteq GRS_{n,k}(\alpha, v)$. Se $f(x)$ è un polinomio, denotiamo con $\mathbf{f}$ la code word che vi è associata. Dato che quest'ultima dipende sia da α che da v, quando sarà necessario utilizzeremo la seguente scrittura, che evita ambiguità:

$$ev_{\alpha,v}(f(x)) = (v_1 f(\alpha_1), v_2 f(\alpha_2), \ldots, v_n f(\alpha_n)) \text{ con } \mathbf{f} = ev_{\alpha,v}(f(x)).$$

Questa notazione ci ricorda che il polinomio $f(x)$ viene calcolato in α e riscalato con v. Mostriamo ora un primo risultato che riguarda i codici GRS.

Teorema 3.2. *Il codice $GRS_{n,k}(\alpha, v)$ è un (n, k) - codice, lineare sul campo F, di lunghezza $n \leq |F|$. Inoltre la distanza di Hamming minima tra due code word è $n - k + 1$ con $k \neq 0$.*

Dimostrazione. Per definizione, le entrate di α sono tutte distinte tra loro, perciò $n \leq |F|$. Se poi $a \in F$ e $f(x), g(x) \in F[x]_k$, allora anche $af(x) + g(x) \in F[x]_k$ e

$$ev_{\alpha,v}(af(x) + g(x)) = aev_{\alpha,v}(f(x)) + ev_{\alpha,v}(g(x)) = a\mathbf{f} + \mathbf{g}.$$

Quindi $GRS_{n,k}(\alpha, v)$ è un codice lineare, di lunghezza n su F.

Siano $f(x), g(x) \in F[x]_k$ polinomi distinti fra loro. Definiamo $h(x) = f(x) - g(x) \neq 0$, che ha grado minore a k. Allora $\mathbf{h} = \mathbf{f} - \mathbf{g}$ e la distanza di Hamming tra $\mathbf{f}$ e $\mathbf{g}$ coincide con il numero di entrate non nulle di $\mathbf{h}$. Questo è dato da n meno il numero di zeri in $\mathbf{h}$. Dato che i v_i sono non nulli, $d(\mathbf{f}, \mathbf{g})$ equivale a n meno il numero di radici che $h(x)$ ha nell'insieme $\{\alpha_1, \ldots, \alpha_n\}$. Dato che $h(x)$ è un polinomio di grado minore di k, avrà al massimo $k-1$ radici; ne consegue che il numero di entrate non nulle di $\mathbf{h}$ è almeno $n-(k-1) = n-k+1$. Perciò la distanza minima di Hamming tra due code words, che denotiamo per brevità con d_{min}, sarà $d_{min} \geq n-k+1$.

L'altro verso della disequazione $d_{min} \leq n-k+1$ segue dal teorema 1.15 visto nella sezione 1.4.

Infine due polinomi distinti $f(x), g(x) \in F[x]_k$ forniscono code words distinte, quindi il codice contiene $|F|^n$ code words e ha dimensione k. □

Cerchiamo una matrice generatrice per un un codice $GRS_{n,k}(\alpha, v)$. Ogni base $f_1(x), f_2(x), \ldots, f_k(x)$ di $F_k[x]$ dà una base $\mathbf{f_1}, \ldots, \mathbf{f_k}$ per il codice. Scegliamo perciò la base di polinomi $\{1, x, \ldots, x^i, \ldots, x^{k-1}\}$, per la quale vale

$$ev_{\alpha.v}(x^i) = (v_1\alpha_1^i, \ldots, v_j\alpha_j^i, \ldots, v_n\alpha_n^i), \text{ con } 0 \leq i \leq k-1.$$

Definizione 3.3. Con le scelte appena effettuate, definiamo la **matrice generatrice canonica** per il codice $GRS_{n,k}(\alpha, v)$:

$$\begin{pmatrix} v_1 & v_2 & \ldots & v_j & \ldots & v_n \\ v_1\alpha_1 & v_2\alpha_2 & \ldots & v_j\alpha_j & \ldots & v_n\alpha_n \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ v_1\alpha_1^i & v_2\alpha_2^i & \ldots & v_j\alpha_j^i & \ldots & v_n\alpha_n^i \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ v_1\alpha_1^{k-1} & v_2\alpha_2^{k-1} & \ldots & v_j\alpha_j^{k-1} & \ldots & v_n\alpha_n^{k-1} \end{pmatrix}$$

Risulta particolarmente utile riguardare il Teorema 3.2 in termini di interpolazione polinomiale. Introdurremo dei concetti funzionali alla comprensione dei paragrafi successivi, senza entrare nello specifico di argomenti che non competono questa trattazione e rimandiamo ad altra sede il loro approfondimento.

Diamo per noto il seguente fatto:

Osservazione. Un polinomio di grado minore o uguale a k è univocamente determinato dal suo valore in k (o più) punti distinti.

Ogni code word con k entrate nulle corrisponde ad un polinomio di grado minore di k che si annulla in k punti distinti e che quindi deve essere il polinomio nullo. Data quindi una $n-upla$ $\mathbf{f}$, possiamo facilmente ricostruire l'unico polinomio $f(x)$ di grado minore di n per il quale $\mathbf{f} = ev_{\alpha,v}(f(x))$.

Utilizzeremo le seguenti notazioni:

$$L(x) = \prod_{i=1}^{n}(x - \alpha_i) \text{ e } L_i(x) = \frac{L(x)}{(x - \alpha_i)} = \prod_{j \neq i}(x - \alpha_j).$$

I polinomi $L(x)$ e $L_i(x)$ sono monici, di grado n ed $n-1$ rispettivamente. Il vettore codificato $\mathbf{f}$ ha come i-esima coordinata $v_i f(\alpha_i)$, pertanto abbiamo abbastanza informazioni per calcolare $f(x)$ utilizzando la *formula di interpolazione di Lagrange*:

$$f(x) = \sum_{i=1}^{n} \frac{L_i(x)}{L_i(\alpha_i)} f(\alpha_i), \text{ con } L_i(\alpha_i) \neq 0 \text{ sempre .}$$

Nel presentare i GRS codici abbiamo visto come essi possano essere definiti tramite la loro matrice generatrice canonica (definizione 3.3). Abbiamo però già notato nella sezione 1.4 quanto possa essere utile parlare di un (n,k) - codice lineare in termini della sua matrice di controllo. In particolare questo ci apre alla possibilità di sfruttare la decodifica tramite le sindromi.

Diamo ora la definizione di *codice duale*, e presentiamo alcune proprietà di questi ultimi, che serviranno successivamente a dimostrare la FORMULAZIONE DI GOPPA PER I GRS CODICI, ovvero il teorema 3.7.

Definizione 3.4. Sia C un codice, non necessariamente lineare, su F^n, con F un campo. Il **codice duale** di C, denotato con $C^\perp$, è il codice

$$C^\perp = \{x \in F^n \mid x \cdot c = 0, \text{ per ogni } c \in C\},$$

dove $x \cdot c$ è l'usuale prodotto scalare. Inoltre $C^\perp$ è lineare anche se C non lo è.

Sia G la matrice generatrice del codice C. Allora $x \in C^\perp$ se e solo se ${}^t xG = 0$, dato che le colonne di G sono una base per il sottospazio dei vettori codificiati, come osservato precedentemente nella sezione 1.4. Enunciamo, senza provare, il seguente lemma:

Lemma 3.5. *Se C è un (n,k) - codice lineare sul campo F, allora il suo codice duale $C^\perp$ è un $(n, n-k)$ - codice lineare sul campo F.*

Con il seguente teorema, esplicitiamo la relazione che c'è tra un GRS codice ed il suo duale, che per il lemma 3.5 appena visto sarà un codice lineare a sua volta. Ricordiamo anche che un GRS codice è lineare sul campo F per il teorema 3.2.

Teorema 3.6. *Vale*

$$GRS_{n,k}(\alpha, v)^\perp = GRS_{n,n-k}(\alpha, u),$$

con $u = (u_1, \ldots, u_n)$ e $u_i^{-1} = v_i \prod_{j \neq i}(\alpha_i - \alpha_j)$.

Dimostrazione. Con le notazioni introdotte precedentemente $u_i = v_i^{-1} L_i(\alpha_i)^{-1}$. Vogliamo dimostrare che ogni $\mathbf{f}$ in $GRS_{n,k}(\alpha, v)$ ha prodotto scalare nullo contro ogni $\mathbf{g}$ di $GRS_{n,n-k}(\alpha, u)$; siano quindi $\mathbf{f} = ev_{\alpha,v}(f(x))$ e $\mathbf{g} = ev_{\alpha,u}(g(x))$. Il polinomio $f(x)$ ha grado minore di k, mentre $g(x)$ ha grado minore di $n-k$. Pertanto il loro prodotto $f(x)g(x)$ avrà grado al massimo $n-2$. Utilizzando l'*interpolazione di Lagrange*

$$f(x)g(x) = \sum_{i=1}^{n} \frac{L_i(x)}{L_i(\alpha_i)} f(\alpha_i)g(\alpha_i).$$

Uguagliando i coefficienti del termine x^{n-1} a destra e a sinistra otteniamo:

$$\begin{aligned} 0 &= \sum_{i=1}^{n} \frac{1}{L_i(\alpha_i)} f(\alpha_i) g(\alpha_i) \\ &= \sum_{i=1}^{n} (v_i f(\alpha_i)) \Big(\frac{1}{v_i L_i(\alpha_i)} g(\alpha_i) \Big) \\ &= \sum_{i=1}^{n} (v_i f(\alpha_i))(u_i g(\alpha_i)) \\ &= \mathbf{f} \cdot \mathbf{g}. \end{aligned}$$

□

Per verificare che $\mathbf{f}$ sia una code word di $C = GRS_{n,k}(\alpha, v)$ non è necessario effettuare il prodotto scalare contro ogni $\mathbf{g}$ di $C^{\perp} = GRS_{n,n-k}(\alpha, u)$. É infatti sufficiente considerare una base di $C^{\perp}$: una buona scelta sono le righe della matrice generatrice canonica per $C^{\perp}$, vista in 3.3.

Sia $r = n - k$ e sia $c = (c_1, \ldots, c_n) \in F^n$. Allora

$$\begin{aligned} c \text{ è una code word} &\iff 0 = c \cdot ev_{\alpha,u}(x^j), \text{ per } 0 \leq j \leq r-1 \\ &\iff 0 = \sum_{i=1}^{n} c_i u_i \alpha_i^j, \text{ per } 0 \leq j \leq r-1. \end{aligned}$$

Riscriviamo queste r equazioni in un'unica equazione nell'anello dei polinomi $F[z]$, con z una nuova indeterminata. Il vettore c è una code word se e solo se in $F[z]$ vale :

$$\begin{aligned} 0 &= \sum_{j=0}^{r-1} \Big(\sum_{i=1}^{n} c_i u_i \alpha_i^j \Big) z^j \\ &= \sum_{i=1}^{n} c_i u_i \Big(\sum_{j=0}^{r-1} (\alpha_i z)^j \Big) \end{aligned}$$

I polinomi $1 - \alpha z$ e z^r sono primi tra loro $(mod\, z^r)$, perciò è possibile invertire $1 - \alpha z$ nell'anello $F[z](mod\, z^r)$. Infatti :

$$\frac{1}{1-\alpha z} = \sum_{j=0}^{r-1} (\alpha z)^j \qquad (mod\, z^r),$$

è un troncamento dell'usuale espansione della serie geometrica. Ne segue questo teorema:

Teorema 3.7. *(Formulazione di Goppa per i GRS codici.) Il codice di Reed-Solomon Generalizzato $GRS_{n,k}(\alpha, v)$ sul campo F coincide con l'insieme di tutte le n-uple $c = (c_1, \ldots, c_n) \in F^n$ che soddisfano*

$$\sum_{i=1}^{n} \frac{c_i u_i}{1 - \alpha_i z} = 0, \qquad (mod\, z^r),$$

con $r = n - k$ e $u_i^{-1} = v_i \prod_{j \neq i} (\alpha_i - \alpha_j)$.

3.2 Decodifica dei codici Reed-Solomon Generalizzati

In questa sezione presentiamo un efficiente algoritmo di decodifica, specifico per il codice duale di $GRS_{n,r}(\alpha, u)$, partendo dalla Formulazione di Goppa vista come ultimo teorema 3.7 della sezione 3.1.

Supponiamo che $c = (c_1, \ldots, c_n)$ sia trasmesso, che $p = (p_1, \ldots, p_n)$ sia stato ricevuto e che si sia verificato un errore $e = (e_1, \ldots, e_n)$, cioè $p = c + e$.

Definizione 3.8. Definiamo la **sindrome polinomiale** di p

$$S_p(z) = \sum_{i=1}^{n} \frac{p_i u_i}{1 - \alpha_i z} \qquad (\mathrm{mod}\, z^r).$$

É immediato verificare che

$$S_p(z) = S_c(z) + S_e(z) \qquad (\mathrm{mod}\, z^r),$$

e, come conseguenza del teorema 3.7, dato che c è una code word, si ha che

$$S_p(z) = S_e(z) \qquad (\mathrm{mod}\, z^r).$$

Sia ora $B = \{i \,|\, e_i \neq 0\}$ l'insieme che identifica le posizioni in cui è avvenuto l'errore. Allora

$$S_p(z) = S_e(z) = \sum_{b \in B} \frac{e_b u_b}{1 - \alpha_b z} \qquad (\mathrm{mod}\, z^r).$$

D'ora in avanti ometteremo il pedice e scriveremo $S(z)$ intendendo $S_p(z)$. Facendo denominatore comune, troviamo la cosiddetta Key Equation:

$$\sigma(z)S(z) = \omega(z) \qquad (\mathrm{mod}\, z^r) \tag{3.1}$$

dove

$$\sigma(z) = \sigma_e(z) = \prod_{b \in B} (1 - \alpha_b z)$$

e

$$\omega(z) = \omega_e(z) = \sum_{b \in B} e_b u_b \left(\prod_{a \in B, a \neq b} (1 - \alpha_a z) \right).$$

Il polinomio $\sigma(z)$ è chiamato **error locator polynomial** (polinomio che trova l'errore), mentre il polinomio $\omega(z)$ è il **error evaluator polynomial** (o polinomio di valutazione dell'errore).

Dati i polinomi $\sigma(z) = \sigma_e(z)$ e $\omega(z) = \omega_e(z)$ possiamo ricostruire il vettore dell'errore e. Assumiamo al momento che ogni α_i sia non nullo (se qualche α_i fosse 0 valgono risultati simili). Allora:

$$B = \{b \,|\, \sigma(\alpha_b^{-1}) = 0\}$$

e, per ogni $b \in B$,

$$e_b = \frac{-\alpha_b \omega(\alpha_b^{-1})}{u_b \sigma'(\alpha_b^{-1})}$$

dove la dicitura $\sigma'(z)$ denota la derivata formale di $\sigma(z)$.

Se l'errore $e \neq 0$ ha un numero di entrate non nulle che è al massimo $r/2 = (n-k)/2 = (d_{min}-1)/2$, allora detta $S_e(z) = S(z)$ la sindrome polinomiale, si ha che la coppia di polinomi $\sigma(z) = \sigma_e(z)$ e $\omega(z) = \omega_e(z)$ gode di tre proprietà che riassumiamo nel seguente teorema.

Teorema 3.9. *Dati r e $S(z) \in F[z]$ c'è al massimo una coppia di polinomi $\sigma(z), \omega(z) \in F[z]$ che soddisfa:*

1. $\sigma(z)S(z) = \omega(z) \qquad (mod\, z^r)$;
2. $deg\,(\sigma(z)) \leq r/2$ *e* $deg\,(\omega(z)) < r/2$;
3. $mcd\,(\sigma(z), \omega(z)) = 1$ *e* $\sigma(0) = 1$.

Prima di procedere con la dimostrazione, facciamo alcune osservazioni preliminari.

Osservazione. Il punto *1.* è la Key Equation vista in 3.1. La proprietà numero *2.* è conseguenza dell'assunzione che il numero di entrate non nulle del vettore e sia al massimo $r/2$ e della definizione data sopra dei polinomi $\sigma(z), \omega(z)$. Per quanto riguarda il punto *3.* invece, $\sigma(0) = 1$ segue dalla definizione. Inoltre dato che $\sigma(z)$ ha $\deg(\sigma(z))$ radici distinte, o $\mathrm{mcd}\,(\sigma(z), \omega(z)) = 1$ oppure i due polinomi hanno una radice in comune. Ma per ogni radice α_b^{-1} di $\sigma(z)$ si ha $\omega(\alpha_b^{-1}) \neq 0$ e di conseguenza anche $e_b \neq 0$.

Il metodo di decodifica consiste quindi nel trovare il vettore dell'errore e risolvendo la Key Equation. Per provare il teorema 3.9 faremo uso della seguente proposizione, che presentiamo senza dimostrare e la cui verifica è abbastanza immediata.

Proposizione 3.10. *Assumiamo che $\sigma(z), \omega(z)$ soddisfino le proprietà 1.-3. del teorema 3.9 e che $\sigma_1(z), \omega_1(z)$ soddisfino le proprietà 1. e 2. dello stesso teorema. Allora esiste un polinomio $\mu(z)$ tale che $\sigma_1(z) = \mu(z)\sigma(z)$ e $\omega_1(z) = \mu(z)\omega(z)$.*

Dimostrazione. (teorema 3.9) Se esistesse una seconda coppia di polinomi $\sigma_1(z), \omega_1(z)$ che soddisfa le tre proprietà dell'enunciato, per la proposizione 3.10, esiste anche un polinomio $\mu(z)$ tale che

$$\sigma_1(z) = \mu(z)\sigma(z) \text{ e } \omega_1(z) = \mu(z)\omega(z).$$

Quindi $\mu(z)$ divide $\mathrm{mcd}(\sigma_1(z), \omega_1(z)) = 1$ per *3.*. Perciò $\mu(z) = \mu$ è una costante. Ma

$$1 = \sigma_1(0) = \mu(0)\sigma(0) = \mu \cdot 1 = \mu.$$

Così $\sigma_1(z) = \mu(z)\sigma(z) = \sigma(z)$ e $\omega_1(z) = \mu(z)\omega(z) = \omega(z)$. □

Siamo quindi pronti per affontare la decodifica dei GRS codici utilizzando l'algoritmo di divisione euclidea.

Teorema 3.11. *(Decodifica dei GRS codici tramite l'algoritmo euclideo.) Consideriamo il $GRS_{n,k}(\alpha, v)$ codice sul campo F e sia $r = n - k$. Data la sindrome polinomiale $S(z)$, di grado minore di r, il seguente algoritmo si arresta, producendo i polinomi $\tilde{\sigma}(z)$ e $\tilde{\omega}(z)$:*

Siano $a(z) = z^r$ *e* $b(z) = S(z)$. *Definiamo il passo base:*

$$\begin{aligned} r_{-1}(z) &= a(z) & s_{-1}(z) &= 1 & t_{-1}(z) &= 0 \\ r_0(z) &= b(z) & s_0(z) &= 0 & t_0(z) &= 1. \end{aligned}$$

Al passo i*-esimo, costruiamo il polinomio* $r_i(z) = s_i(z)a(z) + t_i(z)b(z)$, *utilizzando il passo* $(i-1)$*-esimo ed il* $(i-2)$*-esimo nella maniera seguente. Siano* $r_{i-2}(z)$ *e* $r_{i-1}(z)(\neq 0)$, *allora effettuando la divisione tra i due,*

$$r_{i-2}(z) = q_i(z)r_{i-1}(z) + r_i(z) \text{ con } \deg(r_i(z)) < \deg(r_{i-1}(z)).$$

Infine definiamo

$$\begin{aligned} s_i(z) &= s_{i-2}(z) - q_i(z)s_{i-1}(z) \\ t_i(z) &= t_{i-2}(z) - q_i(z)t_{i-1}(z). \end{aligned}$$

Cominciamo da $i = 0$. *Se* $r_i(z) \neq 0$ *al passo* i*-esimo, procediamo al successivo. Quando* $r_i(z) = 0$ *l'algoritmo si arresta (in al massimo* $\deg(b(z))$ *passi) e si dichiara che mcd* $(a(z), b(z))$ *è l'unico multiplo monico di* $r_{i-1}(z)$.

Arrestiamo l'algoritmo al passo j*-esimo, in modo che* $\deg(r_j(z)) < r/2$ *e chiamiamo* $\tilde{\sigma}(z) = t_j(z)$ *e* $\tilde{\omega}(z) = r_j(z)$.

Se si verifica durante la trasmissione un errore e, *di al massimo* $r/2 = (d_{min} - 1)/2$ *entrate non nulle e con sindrome polinomiale* $S_e(z) = S(z)$, *allora* $\hat{\sigma}(z) = \tilde{\sigma}(0)^{-1}\tilde{\sigma}(z)$ *e* $\hat{\omega}(z) = \tilde{\sigma}(0)^{-1}\tilde{\omega}(z)$ *sono i polinomi error locator ed error evaluator per* e.

Dimostrazione. L'algoritmo euclideo riduce il grado di $r_j(z)$ ad ogni passo, quindi l'algoritmo sopra definito si arresta effettivamente in un numero finito di passi. Sia $S_e(z) = S(z)$ ed assumiamo che il vettore e abbia un numero di entrate non nulle $\leq r/2$. Allora i polinomi error locator ed error evaluator $\sigma_e(z) = \sigma(z)$ e $\omega_e(z) = \omega(z)$ soddisfano le proprietà *1.*, *2.*, e *3.* del teorema 3.9. Controlliamo in primo luogo che, per j fissato, la coppia $t_j(z), r_j(z)$ verifichi *1.* e *2.*. La proprietà *1.* è la KEY EQUATION ed è soddisfatta ad ogni passo dell'algoritmo euclideo, visto che:

$$r_j(z) = s_j(z)z^r + t_j(z)S(z). \tag{3.2}$$

Per quanto riguarda la proprietà *2.* invece, per come abbiamo scelto j segue che $\deg(r_j(z)) < r/2$ ed anche che $\deg(r_{j-1}(z)) \geq r/2$. Dunque

$$\begin{aligned} \deg(t_j(z)) + r/2 &\leq \deg(t_j(z)) + \deg(r_{j-1}(z)) \\ &= \deg(a(z)) = \deg(z^r) = r. \end{aligned}$$

Da cui ricaviamo che $\deg(t_j(z)) \leq r/2$ e con ciò si conclude la verifica di *2.*.

Per la proposizione 3.10 esiste quindi un polinomio non nullo $\mu(z)$ tale che

$$t_j(z) = \mu(z)\sigma(z) \text{ e } r_j(z) = \mu(z)\omega(z).$$

Sostituiamo quanto appena ottenuto per $t_j(z)$ e $r_j(z)$ nell'equazione 3.2 ed otteniamo

$$\mu(z)\omega(z) = s_j(z)z^r + \mu(z)\sigma(z)S(z)$$

che diventa

$$\mu(z)\big(\omega(z) - \sigma(z)S(z)\big) = s_j(z)z^r.$$

Per la KEY EQUATION 3.1, la parentesi di sinistra è $p(z)z^r$ per un opportuno polinomio $p(z)$. Ci siamo ridotti quindi a considerare $\mu(z)p(z)z^r = s_j(z)z^r$, anzi, semplificando, a

$$\mu(z)p(z) = s_j(z).$$

Perciò $\mu(z)$ divide $\mathrm{mcd}\,(t_j(z), s_j(z)) = 1$, quindi $\mu(z) = \mu$ costante non nulla. Inoltre

$$t_j(0) = \mu(0)\sigma(0) = \mu \cdot 1 = \mu.$$

Ne segue che

$$\sigma(z) = t_j(0)^{-1}t_j(z) \text{ e } \omega(z) = t_j(0)^{-1}r_j(z),$$

come nell'enunciato del teorema. □

Quando utilizziamo questo algoritmo, abbiamo problemi di decodifica in due casi:

- se $\hat{\sigma}(z)$ non si fattorizza in fattori lineari le cui radici sono α_i^{-1}, con molteplicità 1 (ricordiamo che stiamo considerando il caso $\alpha_i \neq 0$ per ogni i);
- se $t_j(0) = 0$, non possiamo effettuare l'ultima divisione per trovare $\hat{\sigma}(z)$.

Comunque il numero di radici di $\hat{\sigma}(z)$ tra le α_i^{-1} deve essere pari al grado di $\hat{\sigma}(z)$ e se ciò non si verifica, abbiamo trovato un errore che non siamo in grado di correggere. Invece se $t_j(0) \neq 0$ e $\hat{\sigma}(z)$ si fattorizza come appena descritto, possiamo valutare l'errore in ogni entrata e ricostruire il vettore errore e con al massimo $r/2$ entrate non nulle. In questo caso, o abbiamo decodificato correttamente la parola ricevuta, o erano presenti più di $r/2$ errori e c'è stato dunque un errore di decodifica.

Assumendo ora r pari ed α un vettore in cui ogni entrata è non nulla, questo algoritmo produce solo error vectors con $r/2$ o meno entrate diverse da 0. Se $\hat{\sigma}(z)$ e $\hat{\omega}(z)$ sono i polinomi che abbiamo trovato applicando il teorema 3.11, questi ci permettono di calcolare un candidato vettore errore e con al massimo $r/2$ entrate non nulle. Per l'*interpolazione di Lagrange* segue che $\hat{\sigma}(z) = \sigma_e(z)$ e $\hat{\omega}(z) = \omega_e(z)$. Inoltre dall'invertibilità modulo z^r di $\sigma(z)$, possiamo ricavare dalla KEY EQUATION $S(z) = S_e(z)$. Pertanto il vettore ricevuto è contenuto in una sfera di raggio $r/2$ di centro una code word e sarà decodificato come quella specifica code word.

Capitolo 4

Alcuni esempi

L'obiettivo di questo ultimo capitolo è quello di presentare alcuni esempi di codici a correzione di errore. Quello che mi propongo, come specificato nell'introduzione, è immaginare di tenere una lezione per degli studenti delle scuole superiori. Si comincerà dunque da alcuni concetti di base riguardanti l'aritmetica modulare, necessari alla comprensione dei due esempi che seguiranno (sezione 4.1). Nelle due sezioni successive 4.2 e 4.3 cercherò di spiegare il funzionamento del Codice Fiscale Italiano e del Codice ISBN-13, due codici con i quali abbiamo a che fare quotidianamente. Va tenuto presente che quanto esposto in questo capitolo è solo una traccia di un possibile lavoro con degli studenti; andrà quindi adeguata di volta in volta, in base alle conoscenze pregresse dei ragazzi e, se del caso, integrata con ulteriori esempi. Non c'è la pretesa di essere troppo formali o completamente esaustivi riguardo la "matematica" presentata nella sezione 4.1, poichè lo scopo principale di questa ipotetica lezione resta comunque quello di suscitare curiosità ed interesse negli ascoltatori nei confronti dell'argomento.

4.1 L'aritmetica modulare

In questa sezione ci preoccuperemo di presentare in modo sintetico e chiaro tutte le nozioni teoriche che serviranno al lettore/ascoltatore per comprendere i due esempi del Codice Fiscale e del codice IBSN-13. Presupporremo che il lettore/ascoltatore conosca l'insieme dei numeri interi $\mathbb{Z}$ e che sappia effettuare divisioni con il resto tra due numeri interi.

Sia $\mathbb{Z}$ l'insieme dei numeri interi.

Definizione 4.1. Un numero $a \in \mathbb{Z}$ si dice **pari** se è divisibile per 2; **dispari** se non lo è.

Ricordiamo anche:

Teorema 4.2. *(L'ALGORITMO DI DIVISIONE.) Per ogni $x, n \in \mathbb{Z}$ con $n > 0$, esistono unici $q, r \in \mathbb{Z}$ tali che*

$$x = n \cdot q + r \ \ e \ 0 \leq r < n.$$

Quindi i numeri pari sono quelli che divisi per 2 danno resto 0, quelli dispari sono quelli che danno resto 1.

Definizione 4.3. Sia ora $n > 1$. Diciamo che due numeri $a, b \in \mathbb{Z}$ sono **congrui modulo n** e scriviamo $a \equiv b \,(\mathrm{mod}\, n)$, o $a \equiv_n b$, se e solo se n divide $a - b$.

Se r è il resto della divisione per n di a, allora $a = q \cdot n + r$ e si ha che $a \equiv_n r$.
La congruenza modulo n è una relazione di equivalenza:

riflessiva $a \equiv_n a$;

simmetrica $a \equiv_n b$ se e solo se $b \equiv_n a$;

transitiva se $a \equiv_n b$ e $b \equiv_n c$, allora $a \equiv_n c$.

Facciamo degli esempi per rendere più chiaro quello di cui stiamo parlando.

Esempio 4.4. Sia $n = 2$. Assegniamo dei valori numerici ad a, b, c appena definiti per verificare "a mano" che valgono queste tre proprietà. Siano $a = 8, b = 4, c = 6$.

Vale $8 \equiv_2 8$ visto che 2 divide $8 - 8 = 0$, cioè la proprietà riflessiva.

Allora $8 \equiv_2 4$ perchè 2 divide $8 - 4 = 4$. Ma vale anche $4 \equiv_2 8$ poichè 2 divide $4 - 8 = -4$. Questa è la proprietà simmetrica.

Infine sappiamo che $8 \equiv_2 4$ e $4 \equiv_2 6$ (infatti 2 divide $4 - 6 = -2$); varrà anche $8 \equiv_2 16$ visto che 2 divide $8 - 16 = -8$, cioè la proprietà che abbiamo chiamato transitiva.

Volendo si può affrontare l'argomento "relazioni" in modo più generale ed approfondito e fornire delle brevi dimostrazioni più formali delle tre proprietà sopracitate, ma questo resta a discrezione dell'insegnante e va relazionato alle conoscenza pregresse degli alunni. Quanto segue non sarà di facile ed immediate comprensione per degli studenti delle scuole superiori, sarà pertanto sviluppato tramite vari esempi.

Il nostro obiettivo è quello di identificare un numero $a \in \mathbb{Z}$ con il resto r dato dalla divisione di a per n. Ad esempio se $n = 3$ identifichiamo 6 con 0 visto che $6 = 2 \cdot 3 + 0$. In questo modo possiamo associare ad ogni numero il suo resto, che sarà compreso tra 0 ed $n - 1$. Diamo due definizioni che verrano esemplificate subito dopo.

Definizione 4.5. Definiamo la **classe di equivalenza modulo n** l'insieme

$$[r] = \{a \in \mathbb{Z} \,|\, a = b \cdot n + r, \exists b \in \mathbb{Z}\}$$

al variare di $0 \leq r \leq n - 1$, cioè l'insieme dei numeri che divisi per n hanno resto r. (Il simbolo $\exists$ vuol dire "esiste"). Chiamiamo l'insieme delle classi di equivalenza

$$\begin{aligned} \mathbb{Z}/n\mathbb{Z} = \mathbb{Z}_n &= \{[0], [1], \ldots, [n-1]\} = \\ &= \{0, 1, \ldots, n-1\} \qquad \text{per semplicità di notazione.} \end{aligned}$$

Cominciamo subito con due brevi esempi.

Esempio 4.6. Riprendiamo l'esempio dei numeri pari e dispari. In questo contesto $n = 2$ e $\mathbb{Z}_2 = \{0, 1\}$. Ogni numero pari corrisponde a 0, mentre ogni numero dispari corrisponde a 1. Infatti $[0] = \{a = b \cdot 2 + 0 = b \cdot 2\}$ è proprio l'insieme dei numeri pari, mentre $[1] = \{a = b \cdot 2 + 1\}$ è l'insieme dei numeri dispari.

Esempio 4.7. Poniamo $n = 10$. Allora $\mathbb{Z}_{10} = \{0, 1, \ldots, 9\}$ ed ogni numero corrisponde alla sua cifra delle unità. Ad esempio $19 = 10 \cdot 1 + 9 \equiv_{10} 9$ e $[9] = \{a = b \cdot 10 + 9\}$.

In $\mathbb{Z}$ abbiamo due operazioni, addizione e moltiplicazione; possiamo definirle anche in $\mathbb{Z}_n$. Basta eseguire l'operazione in $\mathbb{Z}$ e poi ridurre modulo n il risultato. Formalizzando abbiamo la seguente definizione:

Definizione 4.8. Sia $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$. Definiamo la somma (modulo n) $+_n$ ed il prodotto (modulo n) $\cdot_n$ di due numeri interi a, b

$$a +_n b = c \iff a + b \equiv_n c$$
$$a \cdot_n b = c \iff a \cdot b \equiv_n c.$$

Esempio 4.9. In $\mathbb{Z}_2$ si ha che

$$1 +_2 1 = 2 \equiv_2 0.$$

Invece la stessa operazione in $\mathbb{Z}_{10}$ produce

$$1 +_{10} 1 = 2 \equiv_{10} 2.$$

Esempio 4.10. In $\mathbb{Z}_2$ si ha che

$$3 \cdot_2 5 = 15 \equiv_2 1.$$

Invece la stessa operazione in $\mathbb{Z}_{10}$ produce

$$3 \cdot_{10} 5 = 15 \equiv_{10} 5.$$

4.2 Il Codice Fiscale Italiano

In questa sezione cerchiamo di rispondere ad una domanda apparentemente semplice :

Come mai quando digitiamo online il Codice Fiscale
ci viene chiesto di inserirlo una sola volta?

La risposta può sembrare inaspettata: perchè con delle conversioni e delle somme in aritmetica modulo 26 ci si può accorgere se ci sono stati degli errori di digitazione. Ma procediamo con calma ed analizziamo la situazione dall'inizio.

Il Codice Fiscale

Il Codice Fiscale Italiano è composto da 16 caratteri alfanumerici per le persone fisiche e serve ad identificarle in modo univoco. Questi 16 caratteri vengono ricavati nel modo seguente, a partire dai dati anagrafici della persona.

- 3 lettere per il cognome;
- 3 lettere per il nome;
- 2 cifre per l'anno di nascita;
- 1 lettera per il mese di nascita;
- 2 cifre per il giorno di nascita ed il sesso;

- 4 caratteri, 1 lettra e 3 cifre, che identificano il luogo di nascita;
- 1 lettera, che ha funzione **di controllo**.

Le lettere per il cognome e per il nome vengono scelte tra le prime 3 consonanti che occorrono nel cognome e nel nome rispettivamente, prendendo eventualmente una o più vocali qualora non ci fossero abbastanza consonanti. Le 2 cifre per l'anno di nascita sono quelle delle decine e delle unità dell'anno (ad esempio "85" per 1985). Al mese di nascita resta associata una lettera secondo la tabella 4.1

Tabella 4.1: Tabella di conversione del mese di nascita

Gennaio = A	Maggio = E	Settembre = P
Febbraio = B	Giugno = H	Ottobre = R
Marzo = C	Luglio = L	Novembre = S
Aprile = D	Agosto = M	Dicembre = T

Per quanto riguarda data di nascita e sesso le due cifre sono così determinate:

uomini Il giorno di nascita utilizzando i numeri da 1 a 31, con uno 0 se questo non fosse di due cifre;

donne Il giorno di nascita +40, per cui si usano quindi i numeri da 41 a 71.

I 4 caratteri per il luogo di nascita invece sono stabiliti dall'Agenzia del Territorio, incorporata nell' Agenzia delle Entrate dal 2012.

Il sedicesimo carattere è chiamato **carattere alfabetico di controllo** ed è calcolato a partire dai 15 che lo precedono. Viene determinato con l'utilizzo di due tabelle, una per i caratteri che si trovano in posto pari (vedi tabella 4.2) e una per quelli che si trovano in posizione dispari (vedi tabella 4.3). Ogni carattere dei 15 precedenti, viene convertito in un valore numerico da 0 a 25, a seconda della posizione che occupa.

Una volta converiti in valori numerici questi vengono sommati. Il risultato viene poi ridotto *(modulo 26)* e convertito in una lettera con la tabella 4.4. Quindi quando digitiamo il nostro codice fiscale, viene calcolata la lettera di controllo con questo algoritmo. Se quella calcolata con le prime 15 cifre del codice fiscale che abbiamo inserito non corrisponde al sedicesimo carattere che abbiamo scritto, sappiamo che è avvenuto un errore in fase di digitazione.

Vediamo meglio come funziona l'algoritmo sopra descritto con un esempio pratico:

Esempio 4.11. Immaginiamo di voler calcolare il codice fiscale di (una persona fittizia) Maria Rossi, nata a Padova il 13/07/1996. Allora:

- le lettere per il cognome sono RSS;
- le lettere per il nome MRA;
- le due cifre per l'anno di nascita sono 96;
- la lettera per il mese, luglio, è L (utilizzando la tabella 4.1);

- le cifre per il giorno di nascita e il sesso sono $13 + 40 = 53$;
- il codice che identifica il comune di Padova è G224.

Calcoliamo infine il carattere di controllo utilizzando le 3 tabelle 4.2, 4.3 e 4.4. Convertiamo ogni carattere (ad esempio la prima R, che si trova in posto dispari, diventa 8) e sommiamo :

$$8 + 18 + 12 + 12 + 8 + 0 + 21 + 6 + 4 + 5 + 7 + 6 + 5 + 2 + 9 = 123 \equiv_{26} 19.$$

Pertanto l'ultima lettera del codice fiscale di Maria è $19 \to T$.

Tabella 4.2: Tabella di conversione dei caratteri in posizione pari

A o 0 = 0	F o 5 = 5	K = 10	P = 15	U = 20
B o 1 = 1	G o 6 = 6	L = 11	Q = 16	V = 21
C o 2 = 2	H o 7 = 7	M = 12	R = 17	W = 22
D o 3 = 3	I o 8 = 8	N = 13	S = 18	X = 23
E o 4 = 4	J o 9 = 9	O =14	T = 19	Y = 24
				Z = 25

Tabella 4.3: Tabella di conversione dei caratteri in posizione dispari

A o 0 = 1	F o 5 = 13	K = 2	P = 3	U = 16
B o 1 = 0	G o 6 = 15	L = 4	Q = 6	V = 10
C o 2 = 5	H o 7 = 17	M = 18	R = 8	W = 22
D o 3 = 7	I o 8 = 19	N = 20	S = 12	X = 25
E o 4 = 9	J o 9 = 21	O = 11	T = 14	Y = 24
				Z = 23

Tabella 4.4: Tabella per la determinazione del carattere di controllo

0 = A	5 = F	10 = K	15 = P	20 = U
1 = B	6 = G	11 = L	16 = Q	21 = V
2 = C	7 = H	12 = M	17 = R	22 = W
3 = D	8 = I	13 = N	18 = S	23 = X
4 = E	9 = J	14 = O	19 = T	24 = Y
				25 = Z

4.3 Il codice ISBN-13

Concludiamo questa "lezione" parlando del codice **ISBN-13**. L'ISBN-13, o *International Standard Book Number* è una sequenza numerica di 13 cifre, utilizzata a livello internazionale per identificare univocamente i libri; fino al 2007

erano 10 cifre e prendeva il nome di ISBN-10. Nel codice sono presenti informazioni quali l'area linguistica dell'editore, la casa editrice ed il titolo del libro; viene attribuito un codice ISBN ad ogni edizione di un libro (tranne le ristampe) e una volta assegnato questo non può più essere riutilizzato. Ad esempio un e-book, un libro a copertina rigida ed uno a copertina "morbida" che siano la stessa edizione di uno stesso libro avranno codici ISBN diversi.

La domanda che possiamo porci é:

Cosa accomuna il Codice Fiscale ed il codice ISBN?

La risposta si cela nell'ultima cifra: infatti anche il codice ISBN, come il codice fiscale, possiede una **cifra di controllo**, sempre in ultima posizione, calcolata a partire dalle 12 precedenti. Grazie a questa check digit infatti siamo in grado di capire se si è verificato un errore in fase di digitazione.

Definizione 4.12. Chiamiamo codici come questi **error detecting codes**, cioè codici che trovano gli errori (in questo caso del codice ISBN-13 uno).

Vediamo quindi come funziona il codice ISBN-13. Le 13 cifre sono così distribuite:

Prefisso EAN Le prime 3 cifre indicano che si tratta di un libro, per ora sono 978 o 979;

Gruppo linguistico Sono 1-5 cifre che identificano il paese o l'area lingiustica dell'editore;

Editore Sono 2-7 cifre che codificano la casa editrice o il marchio editoriale;

Titolo Sono 1-6 cifre che individuano il libro;

Cifra di controllo Una sola cifra, che può essere un numero da 0 a 9.

Come notiamo dall'elenco, solo la prima e l'ultima parte hanno una lunghezza predefinita, mentre le altre parti variano in modo complementare fra loro. Si usa quindi separare le diverse parti con degli spazi o dei trattini, altrimenti risalire all'ISBN-13 corretto sarebbe molto complicato. I tre settori centrali hanno a disposizione un totale di 9 cifre: ciò comporta che meno cifre si utilizzano per il gruppo linguistico e per l'editore e più se ne hanno a disposizione per il titolo. Il codice è così strutturato per permettere alle lingue ed agli editori che pubblicano più libri di poter utilizzare più codici titolo. L'ultima cifra, come già accennato, ha funzione di controllo e viene calcolata in modo molto semplice a partire dalle altre con delle somme pesate e delle riduzioni *modulo 10*.

Supponiamo che il nostro codice sia la stringa numerica $a_1a_2\dots a_{13}$ con $a_i \in \mathbb{Z}_{10}$ e calcoliamo la cifra di controllo a_{13}. Pesiamo gli a_i moltiplicandoli per 1 se si trovano in posizione dispari, per 3 se si trovano in posto pari. Sommiamo quanto ottenuto e riduciamo modulo 10, trovando così un numero compreso tra 0 e 9, sia questo b. Chiamiamo $r = 10 - b$. Se $r < 10$ allora $a_{13} = r$; se invece $r = 10$ (nel caso quindi in cui $b = 0$) allora $a_{13} = 0$.

$$
\begin{aligned}
r &= 10 - b = \\
&= 10 - \Big[\Big(\sum_{i=2k+1} a_{2k+1} \quad + \sum_{i=2k} 3a_{2k}\Big) (\mathrm{mod}\, 10)\Big] = \\
&= 10 - [(1 \cdot a_1 + 3 \cdot a_2 + 1 \cdot a_3 + 3 \cdot a_4 + 1 \cdot a_5 + 3 \cdot a_6 + 1 \cdot a_7 + 3 \cdot a_8 \\
&+ 1 \cdot a_9 + 3 \cdot a_{10} + 1 \cdot a_{11} + 3 \cdot a_{12}) (\mathrm{mod}\, 10)]
\end{aligned}
$$

Ed allora

$$a_{13} = \begin{cases} r, & \text{se } r < 10; \\ 0, & \text{se } r = 10. \end{cases}$$

Un codice ISBN-13 è valido se soddisfa:

$$\big[1 \cdot a_1 + 3 \cdot a_2 + 1 \cdot a_3 + 3 \cdot a_4 + 1 \cdot a_5 + 3 \cdot a_6 + 1 \cdot a_7 + \\ + 3 \cdot a_8 + 1 \cdot a_9 + 3 \cdot a_{10} + 1 \cdot a_{11} + 3 \cdot a_{12} + 1 \cdot a_{13}\big] (\mathrm{mod}\, 10) = 0. \quad (4.1)$$

Dato un codice ISBN-13 effettuando questo controllo possiamo capire se è stato trascritto correttamente o se c'è al più una cifra sbagliata. Questo algoritmo non rileva però alcuni tipi di trasposizione, cioè quando due cifre vengono scambiate di posto.Una di queste è il caso di due cifre adiacenti scambiate e la cui differenza è 5, ad esempio supponiamo che 61 sia stato digitato come 16. Se l'ordine corretto contribuisce con $3 \cdot 6 + 1 \cdot 1 = 19$ nel calcolo di b, l'ordine "sbagliato" 16 contribuisce a b con $3 \cdot 1 + 1 \cdot 6 = 9$. Ma $19 \equiv_{10} 9$ quindi il risultato finale non cambia e le cifre di controllo dei due codici, quello corretto e quello sbagliato, sono uguali.

Concludiamo con un esempio, in cui verifichiamo che un codice ISBN-13 sia stato trascritto correttamente.

Esempio 4.13. Consideriamo il libro *Harry Potter e i Doni della Morte* [5], pubblicato per la prima volta in Italia nel 2008; il suo codice ISBN-13 è 978-88-8451-878-1. Vediamo se soddisfa la condizione 4.1.

$$1 \cdot 9 + 3 \cdot 7 + 1 \cdot 8 + 3 \cdot 8 + 1 \cdot 8 + 3 \cdot 8 + 1 \cdot 4 + 3 \cdot 5 + \\ + 1 \cdot 1 + 3 \cdot 8 + 1 \cdot 7 + 3 \cdot 8 + 1 \cdot 1 \,(\mathrm{mod}\, 10) = \\ = 170 \equiv_{10} 0.$$

Quindi il codice scritto è effettivamente un codice ISBN, e 1 è la sua cifra di controllo.

Bibliografia

[1] Agenzia delle Entrate. *Informazioni sulla codificazione delle persone fisiche*. Online; in data 2-dicembre-2023. 1997. URL: https://www.agenziaentrate.gov.it/portale/web/guest/schede/istanze/richiesta-ts_cf/informazioni-codificazione-pf.

[2] W. J. Gilbert e W. K. Nicholson. *Modern algebra with applications*. 2nd ed. Pure and applied mathematics. Hoboken, N.J: Wiley-Interscience, 2004. ISBN: 1-280-34418-0.

[3] J. I. Hall. *Notes on coding theory*. Note di un corso universitario. Michigan State University, 2001–2015. URL: https://users.math.msu.edu/users/halljo/classes/codenotes/coding-notes.html.

[4] Agenzia Italiana ISBN. *Cosa è il codice ISBN?* Online; in data 3-dicembre-2023. URL: https://www.isbn.it/CodiceISBN.aspx.

[5] J. K. Rowling. *Harry Potter e i Doni della Morte*. Harry Potter. Salani, 2008. ISBN: 978-88-8451-878-1.