Teoria dell'informazione e della trasmissione

Indice

1. Lezione 01 .... **2**
1.1. Introduzione 2
1.1.1. Storia 2
1.1.2. Shannon vs Kolmogorov 2
1.1.3. Obiettivi di Shannon 2
1.1.4. Primo teorema di Shannon 3
1.1.5. Secondo teorema di Shannon 4
1.2. Codifica della sorgente 5
1.2.1. Introduzione matematica 5
1.2.2. Prima applicazione 5
1.2.3. Modello statistico 6
2. Lezione 02 .... **7**
2.1. Codici 7
2.1.1. Codice univocamente decodificabile 7
2.1.2. Codice istantaneo 7
2.1.3. Gerarchia dei codici 8
2.2. Disuguaglianza di Kraft 9
2.2.1. Definizione 9
2.2.2. Applicazione 10

1. Lezione 01

1.1. Introduzione

1.1.1. Storia

La **teoria dell'informazione** nasce nel 1948 grazie a **Claude Shannon** (1916-2001), un impiegato alla "Telecom" americana al quale sono stati commissionati due lavori: data una comunicazione su filo di rame, si voleva sfruttare tutta la capacità del canale, ma al tempo stesso correggere gli errori di trasmissione dovuti al rumore presente.

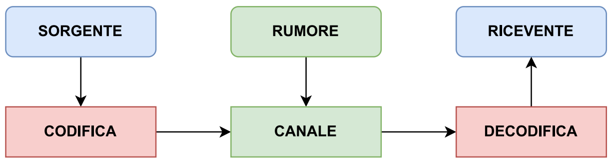
Nel luglio 1948 infatti viene pubblicato l'articolo "*A Mathematical Theory of Communication*" da parte di Bell Labs, dove Shannon pone le basi della teoria dell'informazione.

Ma non è l'unico personaggio che lavora in questo ambito: infatti, nel 1965, tre matematici russi pubblicano degli articoli che vanno a perfezionare il lavoro fatto anni prima da Shannon.

I tre matematici sono Gregory Chaitin (1947-), Ray Solomonoff (1926-2009) e **Andrey Kolmogorov** (1903-1987), ma si considerano solo gli articoli di quest'ultimo poiché ai tempi era molto più famoso dei primi due.

1.1.2. Shannon vs Kolmogorov

La situazione generica che troveremo in quasi la totalità dei nostri studi si può ridurre alla comunicazione tra due entità tramite un **canale affetto da rumore**.



Quello che distingue Shannon da Kolmogorov è l'approccio: il primo propone un **approccio ingegneristico**, ovvero tramite un modello formato da una distribuzione di probabilità si va a definire cosa fa *in media* la sorgente, mentre il secondo propone un **approccio rigoroso e formale**, dove sparisce la nozione di media e si introduce la nozione di sorgente in modo *puntuale*.

In poche parole, dato un messaggio da comprimere:
- Shannon direbbe "lo comprimo *in media* così, e lo comprimerei così anche se il messaggio fosse totalmente diverso";
- Kolmogorov direbbe "lo comprimo *esattamente* così, ma lo comprimerei in modo totalmente diverso se il messaggio fosse diverso".

1.1.3. Obiettivi di Shannon

Gli obiettivi che Shannon vuole perseguire sono due:
- **massimizzare** l'informazione trasmessa *ad ogni utilizzo del canale*;
- **minimizzare** il numero di errori di trasmissione dovuti alla presenza del rumore nel canale.

La parte "*ad ogni utilizzo del canale*" viene inserita per dire che, ogni volta che si accede al canale, deve essere utilizzato tutto, mentre senza questa parte una sorgente potrebbe mandare l'1% del messaggio ad ogni accesso al canale, mandandolo sì tutto ma senza sfruttare a pieno la banda.

Shannon risolverà questi due problemi con due importantissimi teoremi:

- $I°$ **teorema di Shannon**, che riguarda la *source coding*, ovvero la ricerca di un codice per rappresentare i messaggi della sorgente che massimizzi l'informazione spedita sul canale, ovvero massimizzi la sua **compressione**;
- $II°$ **teorema di Shannon**, che riguarda la *channel coding*, ovvero la ricerca di un codice per rappresentare i messaggi della sorgente che minimizzi gli errori di trasmissione dovuti alla presenza del rumore nel canale.

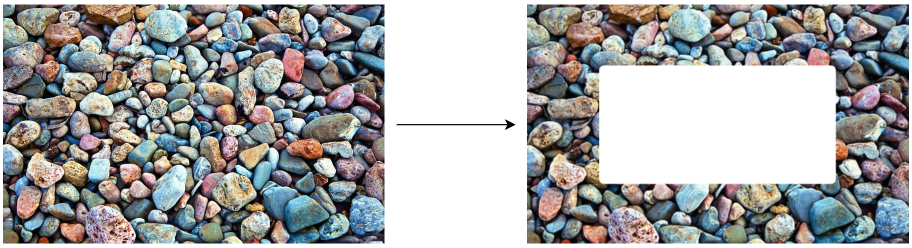
L'approccio che viene usato è quello *divide-et-impera*, che in questo caso riesce a funzionare bene e riesce ad unire i risultati dei due teoremi di Shannon grazie al **teorema di codifica congiunta sorgente-canale** e ad alcune relazioni che legano i due problemi descritti.

In un caso generale del *divide-et-impera* si ricade in una soluzione sub-ottimale.

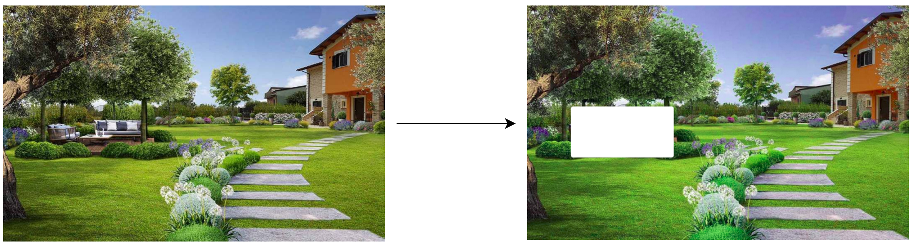
1.1.4. Primo teorema di Shannon

Il primo problema da risolvere è il seguente: come è distribuita l'informazione all'interno di un documento?

Vediamo due esempi dove un documento viene spedito su un canale e alcune informazioni vengono perse per colpa del rumore presente nel canale.



In questo primo esempio notiamo che, nonostante l'informazione persa sia sostanziosa, possiamo in qualche modo "risalire" a quello perso per via delle informazioni che troviamo "nelle vicinanze".



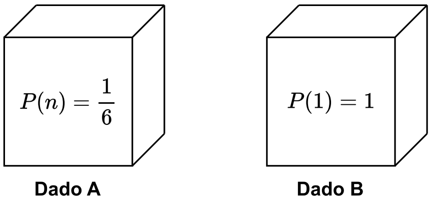
In questo secondo esempio notiamo invece che, nonostante l'informazione persa sia molto meno rispetto a quella precedente, "risalire" al contenuto perso è molto più difficile.

Questi due esempi dimostrano come l'informazione contenuta in un documento **non** è uniforme, e quindi che una distorsione maggiore non implica una perdita maggiore di informazioni.

L'obiettivo del primo teorema di Shannon è eliminare le informazioni inutili e ridondanti, comprimendo il messaggio per poter utilizzare il canale per inviare altre informazioni.

Quello che facciamo è concentrare l'informazione, rendendola **equamente distribuita**, quindi impossibile da ridurre ancora e contenente solo informazioni importanti.

Vediamo un altro esempio: supponiamo di avere due dadi a sei facce, uno *normale* e uno *truccato*, e supponiamo di tirarli assieme per un numero di volte molto grande. Quale dei due dadi mi dà più informazioni?



La risposta è quello *normale*: nel lungo il dado *normale* si "normalizzerà" su una probabilità di circa $\frac{1}{6}$ per ogni possibile faccia, quindi è una sorta di evento **regolare** che mi dà tanta informazione, mentre quello truccato posso sapere già cosa produrrà, quindi è una sorta di evento **prevedibile** che mi dà poca informazione.

In poche parole, massimizzo la probabilità di "ottenere" informazioni se le probabilità di estrarre un simbolo sono uguali.

1.1.5. Secondo teorema di Shannon

Il secondo teorema di Shannon è quello più rognoso, perché si occupa della *channel coding*, ovvero di una codifica che permetta di minimizzare l'informazione persa durante la trasmissione.

Vogliamo questo perché l'informazione che passa sul canale è compressa, quindi qualsiasi bit perso ci fa perdere molte informazioni, non essendoci ridondanza.

Quello che viene fatto quindi è aggiungere **ridondanza**, ovvero più copie delle informazioni da spedire così che, anche perdendo un bit di informazione, lo si possa recuperare usando una delle copie inviate.

La ridondanza che aggiungiamo però è **controllata**, ovvero in base al livello di distorsione del canale utilizzato si inviano un certo numero di copie.

In un **canale ideale** la ridondanza è pari a 0, mentre per canali con rumore viene usata una matrice stocastica, che rappresenta la distribuzione probabilistica degli errori.

IN/OUT	a	b	c	d	e
a	0.7	0.0	0.1	0.1	0.1
b	0.2	0.8	0.0	0.0	0.0
c	0.1	0.0	0.6	0.2	0.1
d	0.0	0.0	0.2	0.5	0.3
e	0.0	0.0	0.0	0.0	1.0

Ogni riga i rappresenta una distribuzione di probabilità che definisce la probabilità che, spedito il carattere i, si ottenga uno dei valori j presenti nelle colonne. Ovviamente, la somma dei valori su ogni riga è uguale a 1.

Se il canale è ideale la matrice risultante è la matrice identità.

1.2. Codifica della sorgente

Andiamo a modellare e formalizzare il primo problema con un modello statistico, utilizzando però un approccio *semplice e bello*: assumiamo che le regole di compressione non siano dipendenti dalle proprietà di un dato linguaggio.

Ad esempio, data la lettera "H" nella lingua italiana, ho più probabilità che esca la lettera "I" piuttosto che la lettera "Z" come successiva di "H", ma questa probabilità nel nostro modello non viene considerata.

Il codice *zip* invece prende in considerazione questo tipo di distribuzione statistica dipendente, e infatti è molto più complicato.

1.2.1. Introduzione matematica

Introduciamo una serie di "personaggi" che saranno utili nella nostra modellazione:
- insieme X: insieme finito di simboli che compongono i messaggi generati dalla sorgente;
- messaggio $\overline{x}$: sequenza di n simboli sorgente; in modo formale,

$$\overline{x} = (x_1, ..., x_n) \in \mathrm{X}^n, \text{ con } x_i \in \mathrm{X} \ \forall i \in \{1, ..., n\};$$

- base D;
- insieme $\{0, ..., D-1\}$: insieme finito dei simboli che compongono il codice scelto;
- insieme $\{0, ..., D-1\}^+$: insieme di tutte le possibili parole di codice esprimibili tramite una sequenza non vuota di simboli di codice; in modo formale,

$$\{0, ..., D-1\}^+ = \bigcup_{n=1}^{\infty} \{0, ..., D-1\}^n.$$

  Un altro nome per le parole di codice è sequenze D-arie;
- funzione c: funzione (nel nostro caso *codice*) che mappa ogni simbolo $x \in \mathrm{X}$ in una parola di codice, ovvero una funzione del tipo

$$c : \mathrm{X} \longrightarrow \{0, ..., D-1\}^+.$$

  Questa funzione va ad effettuare un **mapping indipendente**, ovvero tutto viene codificato assumendo che non esistano relazioni tra due o più estrazioni consecutive.

1.2.2. Prima applicazione

Vogliamo trasmettere sul canale i semi delle carte da poker utilizzando il codice binario.

Andiamo a definire:
- $\mathrm{X} = \{\heartsuit, \diamondsuit, \clubsuit, \spadesuit\}$ insieme dei simboli sorgente;
- $c : \mathrm{X} \longrightarrow \{0, 1\}^+$ funzione di codifica;
- $c(\heartsuit) = 0$;
- $c(\diamondsuit) = 01$;
- $c(\clubsuit) = 010$;
- $c(\spadesuit) = 10$.

La codifica proposta è sicuramente plausibile, ma ha due punti deboli:
- *ambiguità*: se ricevo "010" come possibili traduzioni ho:
  - ♣;
  - ♥♠;
  - ♦♥;
- *pessima compressione*: usiamo tre simboli di codice per codificare ♣ e solo uno per codificare ♥.

Quest'ultimo punto debole viene risolto prima introducendo $l_c(x)$ come lunghezza della parola di codice associata ad $x \in \mathrm{X}$, e poi minimizzando la media di tutte le lunghezze al variare di $x \in \mathrm{X}$.

1.2.3. Modello statistico

Per completare il modello abbiamo bisogno di un'altra informazione: la **distribuzione di probabilità**, che definisce la probabilità con la quale i simboli sorgente sono emessi.

Definiamo quindi la sorgente come la coppia $\langle \mathrm{X}, p \rangle$, dove p rappresenta la probabilità prima descritta.

Aggiungiamo qualche "protagonista" a quelli già prima introdotti:
- funzione P_n: non siamo molto interessati ai singoli simboli sorgente, ma vogliamo lavorare con i messaggi, quindi definiamo

$$P_n(\overline{x}) = P_n(x_1, ..., x_n) = \prod_{i=1}^{n} p(x_i).$$

  Possiamo applicare la produttoria perché Shannon assume che ci sia *indipendenza* tra più estrazioni di simboli sorgente;
- variabile aleatoria $\mathbb{X}$: variabile aleatoria $\mathbb{X} : \mathrm{X} \longrightarrow \mathbb{R}$ che rappresenta un'estrazione di un simbolo sorgente;
- insieme $\mathbb{D}$: già definito in precedenza come $\{0, ..., D-1\}$;
- funzione c: già definita in precedenza, ma che riscriviamo come $c : \mathrm{X} \longrightarrow \mathbb{D}^+$.

Con questo modello, fissando X insieme dei simboli sorgente e D base, vogliamo trovare un codice $c : \mathrm{X} \longrightarrow \mathbb{D}^+$ che realizzi la migliore compressione, ovvero che vada a minimizzare il *valore atteso* della lunghezza delle parole di codice, definito come $\mathbb{E}[l_c] = \sum_{x \in \mathbb{X}} l_c(x)p(x)$.

La strategia che viene utilizzata è quella dell'alfabeto Morse: utilizziamo parole di codice corte per i simboli che sono generati spesso dalla sorgente, e parole di codice lunghe per i simboli che sono generatori raramente dalla sorgente.

Il primo problema che incontriamo è quello di evitare la *codifica banale*: se usassi il codice

$$c(x) = 0/1 \quad \forall x \in \mathrm{X}$$

avrei sì una codifica di lunghezza minima ma sarebbe impossibile da decodificare.

Dobbiamo imporre che il codice c sia **iniettivo**, o *non-singolare*.

2. Lezione 02

2.1. Codici

Come detto nella scorsa lezione, andiamo a considerare dei codici non singolari per risolvere la problematica dei due *codici banali*, che minimizzavano sì il valore atteso delle lunghezze delle parole di codice $l_c(x)$, ma rendevano impossibile la decodifica.

Andiamo ora a raffinare ulteriormente i codici singolari presi in considerazione.

2.1.1. Codice univocamente decodificabile

Come detto nella scorsa lezione, siamo interessati alle parole che vengono generate dalla mia sorgente, e non ai singoli caratteri.

Andiamo a definire:
- $\mathrm{X} = \{\heartsuit, \diamondsuit, \clubsuit, \spadesuit\}$ insieme dei simboli sorgente;
- $c : \mathrm{X} \longrightarrow \{0, 1\}^+$ funzione di codifica;
- $c(\heartsuit) = 0$;
- $c(\diamondsuit) = 01$;
- $c(\clubsuit) = 010$;
- $c(\spadesuit) = 10$.

La codifica c scelta è sicuramente non singolare, però abbiamo delle problematiche a livello di decodifica: infatti, possiamo scrivere 01001 come
- $c(\clubsuit, \diamondsuit)$;
- $c(\diamondsuit, \heartsuit, \diamondsuit)$;
- $c(\heartsuit, \spadesuit, \diamondsuit)$

e lato ricevente questo è un problema, perché "unendo" le singole codifiche non otteniamo una parola che è generabile in modo unico.

Introduciamo quindi l'**estensione** di un codice c: essa è un codice $C : \mathrm{X}^+ \longrightarrow \mathbb{D}^+$ definito come $C(x_1, ..., x_n) = c(x_1)...c(x_n)$ che indica la sequenza ottenuta giustapponendo le parole di codice $c(x_1), ..., c(x_n)$.

L'estensione C di un codice c non eredita in automatico la proprietà di non singolarità di c.

Un codice c è **univocamente decodificabile** quando la sua estensione C è non singolare.

Tramite l'**algoritmo di Sardinas-Patterson** siamo in grado di stabilire se un codice è univocamente decodificabile in tempo $O(mL)$, dove $m = |\mathrm{X}|$ e $L = \sum_{x \in \mathrm{X}} l_c(x)$

2.1.2. Codice istantaneo

I codici univocamente decodificabili sono ottimali? Sto minimizzando al meglio $\mathbb{E}[l_c]$?

Prima di chiederci questo vogliamo creare un codice che rispetti un'altra importante proprietà: permetterci di decodificare *subito* quello che mi arriva dal canale senza dover aspettare tutto il flusso.

Andiamo a definire:
- $\mathrm{X} = \{\heartsuit, \diamondsuit, \clubsuit, \spadesuit\}$ insieme dei simboli sorgente;
- $c : \mathrm{X} \longrightarrow \{0, 1\}^+$ funzione di codifica;
- $c(\heartsuit) = 10$;
- $c(\diamondsuit) = 00$;
- $c(\clubsuit) = 11$;
- $c(\spadesuit) = 110$.

Supponiamo di spedire sul canale la stringa 11000…00, e poniamoci lato ricevente. In base al numero di 0 inviati abbiamo due possibili decodifiche:
- #0 pari: decodifichiamo con ♣♦…♦;
- #0 dispari: decodifichiamo con ♠♦…♦.

Nonostante si riesca perfettamente a decodificare, e questo è dato dal fatto che c è un codice univocamente decodificabile, dobbiamo aspettare di ricevere tutta la stringa per poterla poi decodificare, ma in ambiti come lo *streaming* questa attesa non è possibile.

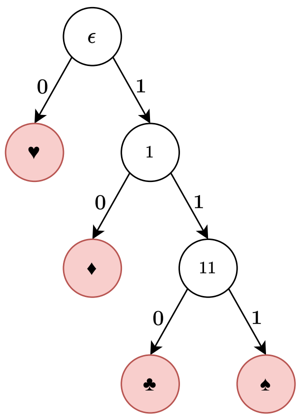
Un altro problema lo riscontriamo a livello di memoria: supponiamo che lato sorgente vengano codificati dei dati dell'ordine dei terabyte, per il ricevente sarà impossibile tenere in memoria una quantità simile di dati per fare la decodifica alla fine della ricezione.

Introduciamo quindi i **codici istantanei**, particolari codici che permettono di decodificare quello che arriva dal canale, appunto, in modo *istantaneo* senza aspettare.

Un codice si dice istantaneo se nessuna parola di codice è *prefissa* di altre.

Andiamo a definire:
- $\mathrm{X} = \{\heartsuit, \diamondsuit, \clubsuit, \spadesuit\}$ insieme dei simboli sorgente;
- $c : \mathrm{X} \longrightarrow \{0, 1\}^+$ funzione di codifica;
- $c(\heartsuit) = 0$;
- $c(\diamondsuit) = 10$;
- $c(\clubsuit) = 110$;
- $c(\spadesuit) = 111$.

Il seguente codice è istantaneo, e lo notiamo osservando l'*albero* che dà origine a questo codice.



Quello che otteniamo è un albero molto sbilanciato.

2.1.3. Gerarchia dei codici

Cerchiamo infine di definire una **gerarchia** tra i codici analizzati fin'ora: se siamo sicuri che i codici non singolari siano sotto-insieme di tutti i codici (*banale*), e che i codici univocamente decodificabili siano sotto-insieme dei codici non singolari (*banale*), siamo sicuri che i codici istantanei siano un sotto-insieme dei codici univocamente decodificabili?

Lemma Se c è istantaneo allora è anche univocamente decodificabile.

Dimostrazione
Andiamo a dimostrare che se c non è univocamente decodificabile allora non è istantaneo. Visto che c non è univocamente decodificabile (supponiamo sia almeno non singolare) esistono due messaggi distinti $x_1, x_2 \in \mathrm{X}^+$ tali che $C(x_1) = C(x_2)$.

Ci sono solo due modi per avere x_1 e x_2 distinti:
1. un messaggio è prefisso dell'altro: se x_1 è formato da x_2 e altri m caratteri, vuol dire che i restanti m caratteri di x_1 devono essere codificati con la parola vuota, che per definizione di codice non è possibile, e soprattutto la parola vuota sarebbe prefissa di ogni altra parola di codice, quindi il codice c non è istantaneo;
2. esiste almeno una posizione in cui i due messaggi differiscono: sia i la prima posizione dove i due messaggi differiscono, ovvero $x_1[i] \neq x_2[i]$ e $x_1[j] = x_2[j]$ per $1 \leq j \leq i - 1$, ma allora $c(x_1[i]) \neq c(x_2[i])$ e $c(x_1[j]) = c(x_2[j])$ perché c è non singolare, quindi sto dicendo che x_1 deve avere x_2 come prefisso (o viceversa), ma, come al punto precedente, otteniamo che c non è istantaneo.

Quindi il codice c non è istantaneo. □

Abbiamo quindi stabilito una gerarchia di questo tipo:

$$\text{codici istantanei} \subset \text{codici univocamente decodificabili} \subset \text{codici non singolari} \subset \text{codici} .$$

Una cosa che possiamo notare è come, aumentando di volta in volta le proprietà di un codice, il valore atteso $\mathbb{E}[l_c]$ che vogliamo minimizzare peggiora, o al massimo rimane uguale: questo perché aggiungendo delle proprietà al nostro codice imponiamo dei limiti che aumentano in modo forzato la lunghezza delle nostre parole di codice.

2.2. Disuguaglianza di Kraft

2.2.1. Definizione

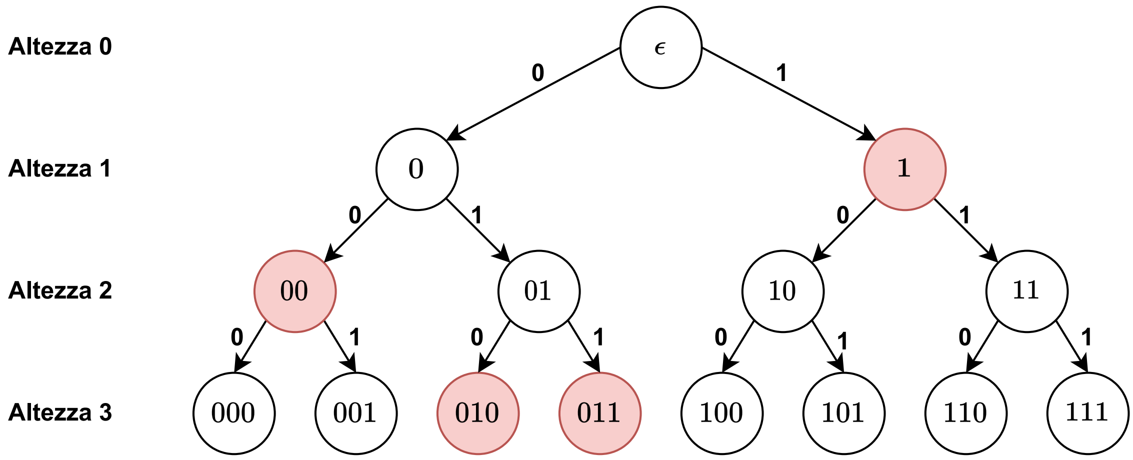
I codici istantanei soddisfano la **disuguaglianza di Kraft**, che ci permette, solo osservando le lunghezze delle parole di codice $l_c(x)$, di dire *se esiste* un codice istantaneo con quelle lunghezze.

Teorema *(Disuguaglianza di Kraft)* Dati $\mathrm{X} = \{x_1, ..., x_n\}$, $D > 1$ e n valori interi positivi $l_1, ..., l_n$, esiste un codice istantaneo $c : \mathrm{X} \to \mathbb{D}^+$ tale che $l_c(x_i) = l_i$ per $i = 1, ..., n$ se e solo se

$$\sum_{i=1}^{n} D^{-l_i} \leq 1.$$

Dimostrazione
($\Longrightarrow$) Sia $l_{\max}$ la lunghezza massima delle parole di c, ovvero $l_{\max} = \max_{i=1,...,n} (l_c(x_i))$. Si consideri l'albero D-ario completo di profondità $l_{\max}$ nel quale posizioniamo ogni parola di codice di c su un nodo dell'albero, seguendo dalla radice il cammino corrispondente ai simboli della parola. Dato che il codice è istantaneo, nessuna parola apparterrà al sotto-albero avente come radice un'altra parola di codice, altrimenti avremmo una parola di codice prefissa di un'altra. Andiamo ora a partizionare le foglie dell'albero in sottoinsiemi disgiunti $A_1, ..., A_n$, dove A_i indica il sottoinsieme di foglie associate alla radice contenente la parola $c(x_i)$.

Nel seguente esempio consideriamo un albero binario di altezza 3, e in rosso sono evidenziate le parole di codice 1, 00, 010, e 011.



Il numero massimo di foglie di un sotto-albero di altezza l_i è $D^{l_{\max}-l_i}$, ma il numero massimo di foglie nell'albero è $D^{l_{\max}}$, quindi

$$\underbrace{\sum_{i=1}^{n}|A_i| = \sum_{i=1}^{n} D^{l_{\max}-l_i}}_{\#\text{foglie coperte}} = \sum_{i=1}^{n} D^{l_{\max}} \cdot D^{-l_i} = D^{l_{\max}} \sum_{i=1}^{n} D^{-l_i} \leq D^{l_{\max}}.$$

Dividendo per $D^{l_{\max}}$ entrambi i membri otteniamo la disuguaglianza di Kraft.

($\Longleftarrow$) Assumiamo di avere n lunghezze positive $l_1, ..., l_n$ che soddisfano la disuguaglianza di Kraft e sia $l_{\max} = \max_{i=1,...,n}(l_i)$ la profondità dell'albero D-ario ordinato e completo usato prima. Associamo ad ogni simbolo $x_i \in \mathrm{X}$ la parola di codice $c(x_1)$, e la inseriamo al primo nodo di altezza l_i che troviamo in ordine lessicografico. Durante l'inserimento delle parole $c(x_i)$ dobbiamo escludere tutti i nodi che appartengono a sotto-alberi con radice una parola di codice già inserita o che includono un sotto-albero con radice una parola di codice già inserita. Il codice così costruito è istantaneo, e visto che rispetta la disuguaglianza di Kraft, la moltiplichiamo da entrambi i membri per $D^{l_{\max}}$ per ottenere

$$\sum_{i=1}^{n} D^{l_{\max}-l_i} \leq D^{l_{\max}},$$

ovvero il numero di foglie necessarie a creare il codice non eccede il numero di foglie disponibili nell'albero. □

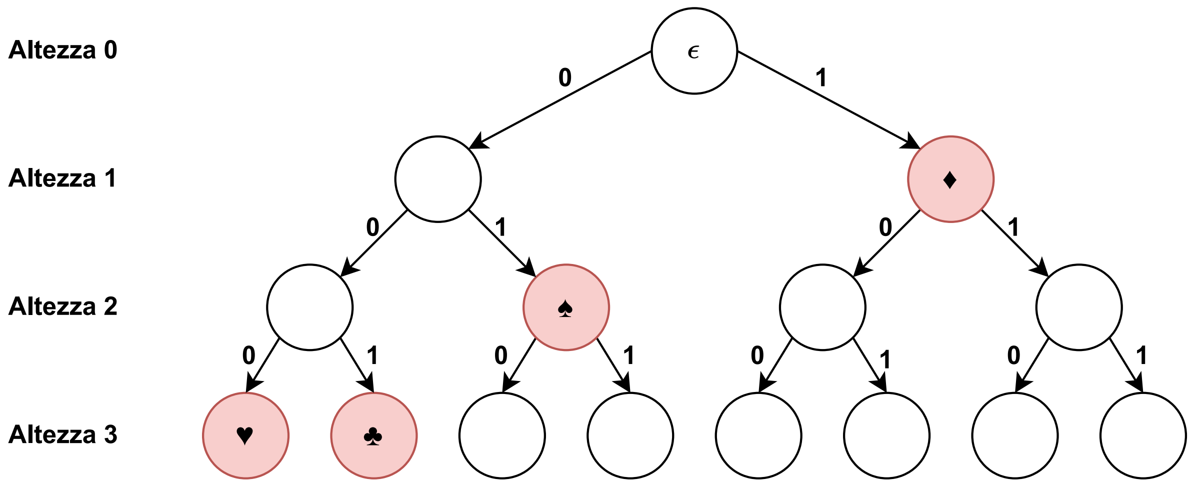
2.2.2. Applicazione

Andiamo a definire:
- $\mathrm{X} = \{\heartsuit, \diamondsuit, \clubsuit, \spadesuit\}$ insieme dei simboli sorgente;
- $c : \mathrm{X} \longrightarrow \{0,1\}^+$ funzione di codifica;
- $l_c(\heartsuit) = 3$;
- $l_c(\diamondsuit) = 1$;
- $l_c(\clubsuit) = 3$;
- $l_c(\spadesuit) = 2$.

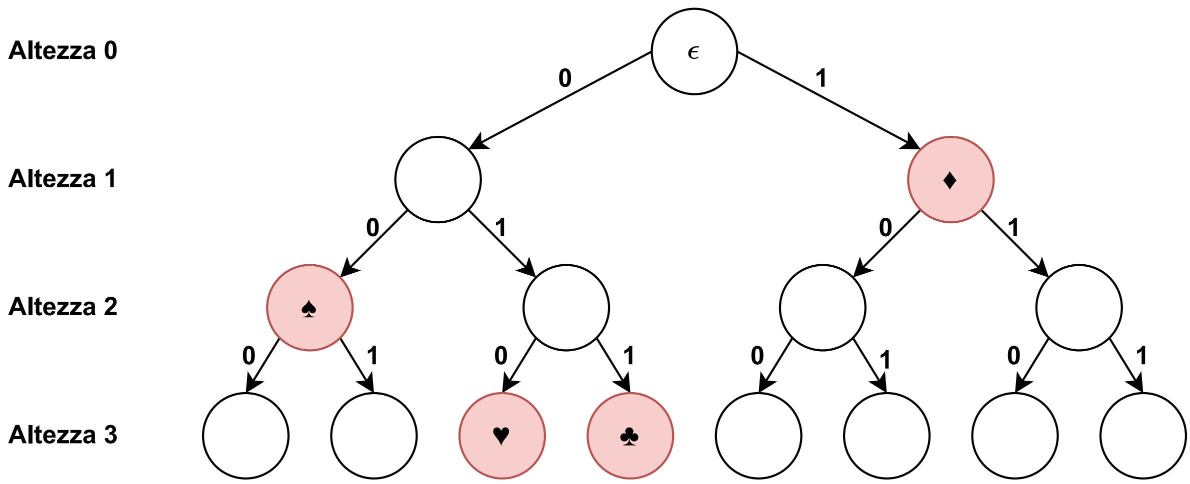
Vogliamo sapere se esiste un codice istantaneo, definito come c, aventi le lunghezze sopra definite. Controlliamo quindi se esse soddisfano la disuguaglianza di Kraft:

$$\sum_{x \in \mathrm{X}} 2^{-l_c(x)} = 2^{-3} + 2^{-1} + 2^{-3} + 2^{-2} = \frac{1}{8} + \frac{1}{2} + \frac{1}{8} + \frac{1}{4} = 1 \leq 1.$$

Andiamo a costruire quindi un codice istantaneo con queste lunghezze costruendo il suo albero.



Il codice ottenuto è l'unico possibile?



Come vediamo, "specchiando" il sotto-albero sinistro con radice ad altezza 1 otteniamo un codice istantaneo diverso dal precedente, ma comunque possibile.

Possiamo concludere quindi che, date n lunghezze positive $l_1, \ldots, l_n$ che soddisfano la disuguaglianza di Kraft, in generale *non è unico* il codice istantaneo che si può costruire.