

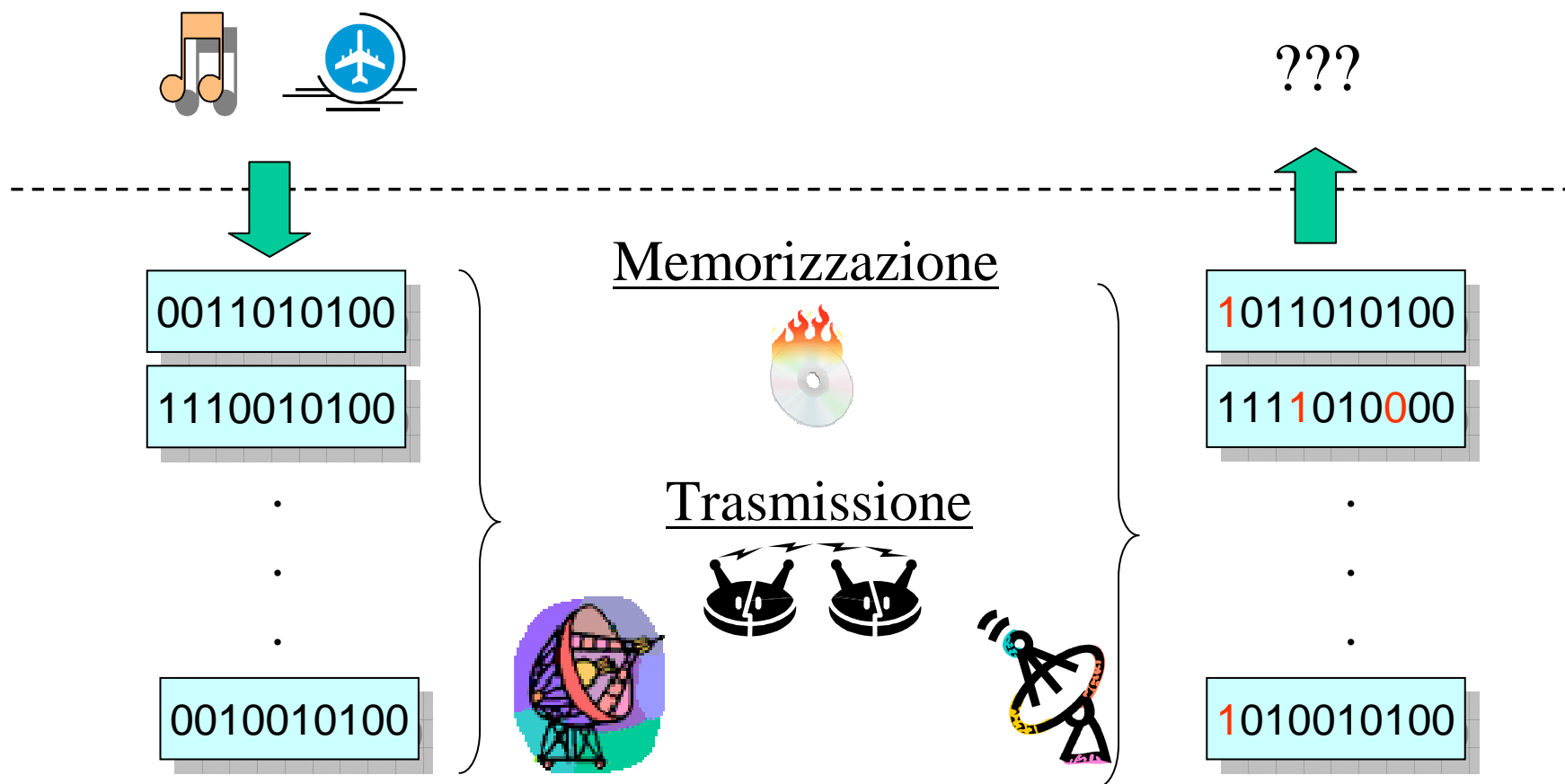
# Tipologie di codici

Nel seguito vedremo tipologie di rappresentazioni diverse:

- Senza assumere limitazioni sul numero di bit a disposizione:  
per numeri [notazione binaria, ovvero posizionale con base 2]
- Disponendo di un numero di bit limitato:
  - numeri naturali
  - interi relativi [valore assoluto e segno, complemento a due]
  - “reali” [virgola fissa e virgola mobile]
  - valori logici, caratteri alfabetici, testi
  - suoni, immagini e sequenze video
  - codici per la rilevazione e correzione di errori
- Codici di compressione (senza | con perdita)

# Codici rivelatori e correttori

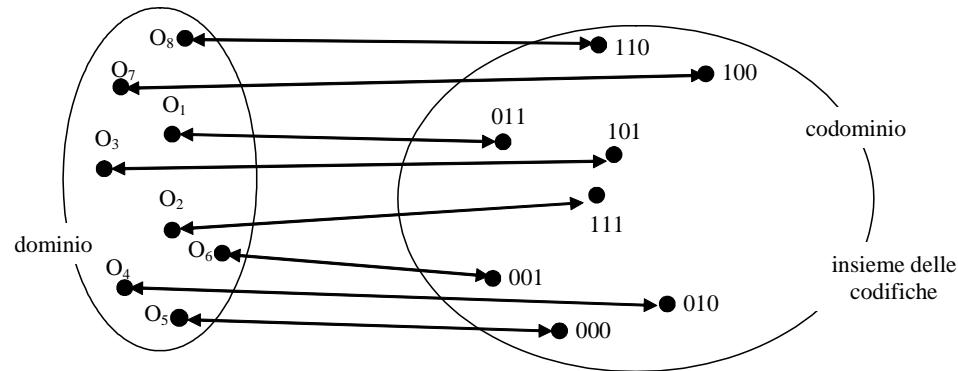
## SCOPO



Se ho errori, mi interessa saperlo (**rivelazione**) o, meglio ancora, correggerli (**correzione**): uso di opportuni codici!

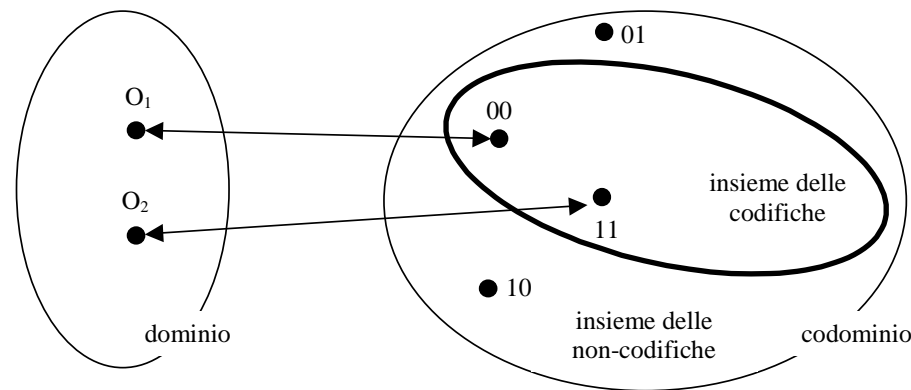
# COME?

## Un esempio di codice non ridondante



*Impossibile  
correggere o  
rivelare errori*

## Un esempio di codice ridondante a due bit



L'insieme delle codifiche è un sottoinsieme proprio del codominio  
 $\Rightarrow$  l'insieme delle non codifiche è non nullo

## QUANTO?

Voglio un codice che, per qualunque codifica sia in grado di rivelare/correggere  $k$  errori comunque siano distribuiti

**Esempio:** un semplice codice per due entità, 5 bit

Atalanta



00000

Brescia



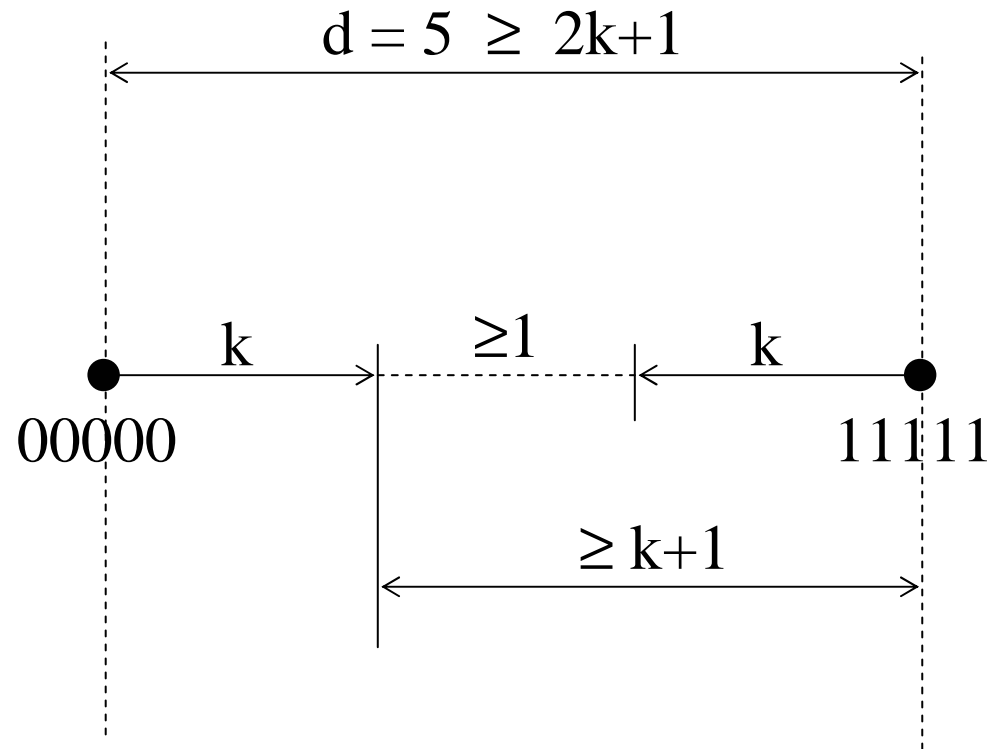
11111



Distanza tra le  
codifiche: 5

- Se ho un numero di errori  $k \leq 4$ : riesco a rivelarli  
 $\Rightarrow d \geq k+1$
- Se ho un numero di errori  $k \leq 2$ : riesco a correggerli  
[es: da 00000 ottengo sequenza con  $d_{00000} \leq 2, d_{11111} \geq 3$ ]  
 $\Rightarrow d \geq 2k+1$

Correzione: l'idea



## QUANTO IN GENERALE?

### **Distanza di un codice:**

minimo valore delle distanze tra ogni coppia di codifiche

### **Rivelazione e correzione di errori**

- Per rivelare  $k$  errori, un codice deve avere distanza  $d$  con  
 $d \geq k+1$
- Per correggere  $k$  errori, un codice deve avere distanza  $d$  con  
 $d \geq 2k+1$

### Esempio di codice rivelatore: il **codice di parità**

$d = 2 \Rightarrow$  può rivelare un errore, correggerne nessuno

## **ESERCIZIO (Appello dell'11 set 2003)**

Qual è la distanza fra le seguenti due codifiche (a 8 bit)

00101000 e 00011001 ?

Un codice con distanza 2 è in grado di rivelare in generale  
due errori? [2]

a)

00101000

00011001

\*\* \*

distanza tra le codifiche = 3

b)

Per rilevare  $k$  errori il codice deve avere distanza  $\geq k+1$ .

Quindi per rilevare 2 errori deve essere

$$d \geq 3$$

mentre un codice con distanza 2 può rivelare al più un errore.



## ESERCIZIO

Per codificare i tre simboli D, F, I si utilizza il seguente codice a 5 bit:

D      →      00000

F      →      11100

I      →      10011

Errori di trasmissione possono dar luogo alla modifica di uno o più bit.

a) Quanti errori è in grado di rivelare il codice in generale?

b) E quanti errori è in grado di correggere?

### Soluzione:

$$d_{12} = 3$$

$$d_{23} = 4$$

$$d_{13} = 3$$

la distanza del codice è pari a 3

a) Errori rivelati:

$d \geq e+1$  quindi  $e_{\max} = 2$  (possono essere rivelati 2 errori)

b) Errori corretti:

$d \geq 2e+1$  quindi  $e_{\max} = (3-1)/2=1$   
(può essere corretto 1 errore)

## ESERCIZIO

Si consideri il codice a 3 valori originari

1  $\rightarrow$  000000

2  $\rightarrow$  000001

3  $\rightarrow$  111111

- a) Trovare quanti errori può correggere e rivelare in generale.
- b) Si supponga di ricevere la sequenza 001111.  
Assumendo che possano essere stati compiuti al più 2 errori, è possibile decodificare correttamente la sequenza?  
Come si giustifica la risposta in relazione al risultato trovato nel punto a)?

## Soluzione

a) Risulta:

$$d_{12} = 1$$

$$d_{23} = 5$$

$$d_{13} = 6$$

quindi la distanza del codice è 1

$\Rightarrow$  il codice non può (in generale) rivelare né tantomeno correggere alcun errore [e rivelati:  $d \geq e+1$ , quindi con  $e=1$  serve  $d \geq 2$ ]

b) Ricevo 001111, mentre avevo:

1 $\rightarrow$	000000 001111	$\text{dist}_1 = 4$
2 $\rightarrow$	000001 001111	$\text{dist}_2 = 3$
3 $\rightarrow$	111111 001111	$\text{dist}_3 = 2$

In questo caso posso decodificare 001111 con il valore “3”. Se sono stati commessi al più due errori, sono sicuro che la decodifica è corretta, perché:

$\text{dist}_1 = 4 > 2$  [000000 non può essere modificato in 001111]

$\text{dist}_2 = 3 > 2$  [000001 non può essere modificato in 001111]

NB: Come mai posso correggere due errori anche se (cfr. punto a) il codice ha distanza 1?

La distanza del codice si riferisce al “caso peggiore” (distanza minima)

⇒ le relative formule garantiscono le proprietà del codice di rilevazione e correzione di errori in ogni caso, ovvero per qualunque simbolo rappresentato e per qualunque posizione degli errori

Per esempio, nel caso venga trasmesso 1 (codificato con 000000), è sufficiente un errore sull'ultimo bit per ottenere 000001, che è pari alla codifica di 2: in tal caso l'errore non verrebbe neppure rivelato!

# Tipologie di codici

Nel seguito vedremo tipologie di rappresentazioni diverse:

- Senza assumere limitazioni sul numero di bit a disposizione:  
per numeri [notazione binaria, ovvero posizionale con base 2]
- Disponendo di un numero di bit limitato:
  - numeri naturali
  - interi relativi [valore assoluto e segno, complemento a due]
  - “reali” [virgola fissa e virgola mobile]
  - valori logici, caratteri alfabetici, testi
  - suoni, immagini e sequenze video
  - codici per la rilevazione e correzione di errori
- **Codici di compressione (senza | con perdita)**

# La compressione dei dati

- Nella codifica digitale, cambiando *modalità di codifica* è possibile ridurre il numero dei bit richiesti
- Vantaggi per *memorizzazione* e *trasmissione* (es: sequenze video)
- Concetti fondamentali validi per qualunque tecnica di compressione:
  - *funzione di compressione*  $F_c$ :  
 $|F_c(S)| < |S|$
  - *rapporto di compressione*:  $|S|/|F_c(S)|$  (NB:  $> 1$ )
  - *funzione di decompressione*  $F_d$ :  
per ricostruire la successione originaria:  $F_d(F_c(S))$
- Due tipi di compressione dei dati:
  - *senza perdita (lossless)*  
è garantito che  $F_d(F_c(S)) = S$ , ovvero  $F_d = F_c^{-1}$
  - *con perdita (lossy)*  
in generale  $F_d(F_c(S)) \neq S$  (perdita di informazione)

# Algoritmi di compressione dati senza perdita

- Si adottano quando non si può perdere informazione, es. programmi in formato eseguibile, file doc
- Svantaggio: ridotto rapporto di compressione
- Principio fondamentale: sottosequenze di bit frequenti sostituite con codici opportuni
- Esempi:
  - formati **ZIP** e **RAR** (con un apposito programma di utilità - es. winzip - si può creare un archivio in cui i file vengono memorizzati in formato compresso)
  - formato **GIF** (per le immagini raster): utilizzo di un “dizionario” con le configurazioni di valori che si ripetono



# Algoritmi di compressione dati con perdita

- Si applicano a dati che hanno origine nel mondo analogico (suoni, immagini, sequenze video, ecc.)
- Si adottano quando si è disposti a perdere una parte dell'informazione durante la compressione: compromesso qualità-rapporto di compressione
- Tecniche dipendenti dalla natura del segnale considerato:
  - per i suoni, variazioni di volume e frequenza al di sotto di una certa soglia non sono percepite dall'orecchio umano + suoni a basso volume sovrapposti a suoni di volume maggiore sono poco udibili (MP3)
  - per le immagini, lievi cambiamenti di colore in punti contigui non sono percepiti dall'occhio umano (JPEG)
  - nelle animazioni video, immagini successive hanno spesso molte parti uguali (uso di vettori di moto) e nel dominio delle frequenze l'occhio umano non è sensibile in modo uniforme (MPEG)