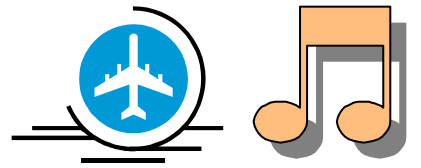


Tipologie di codici

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)

UN ESEMPIO: MEMORIZZAZIONE DI INFORMAZIONI



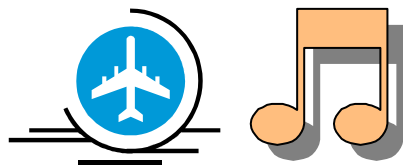
Informazioni da memorizzare
(ad esempio: in un CD)

0011010100

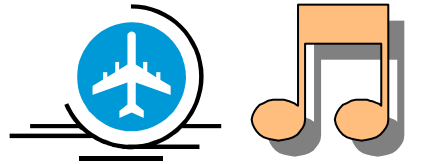
1110010100

•
•
•

0010010100



UN ESEMPIO: MEMORIZZAZIONE DI INFORMAZIONI



Informazioni da memorizzare
(ad esempio: in un CD)

1011010100

1111010000

.

.

.

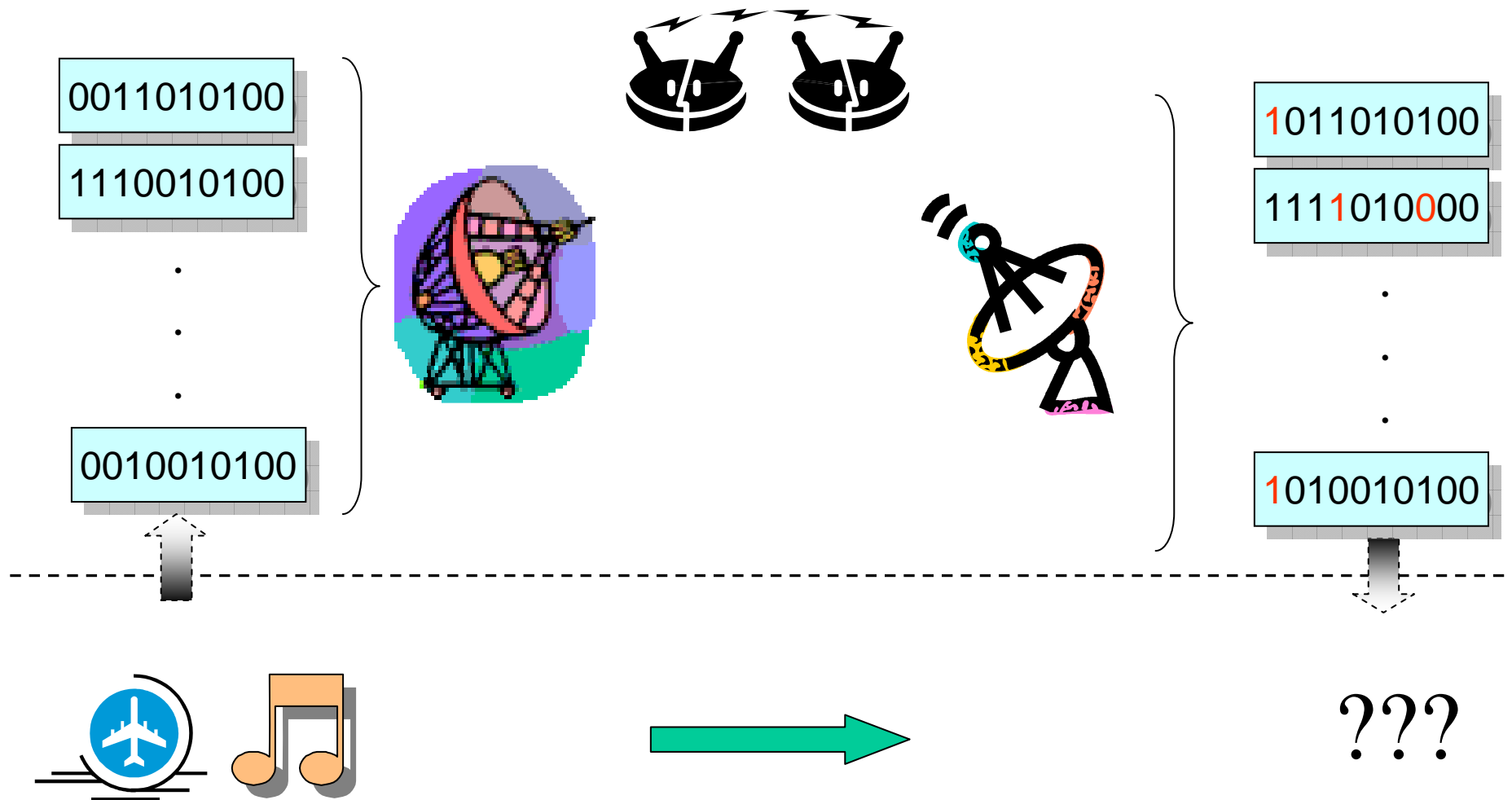
1010010100



???



UN ALTRO ESEMPIO: TRASMISSIONE DI INFORMAZIONE



IL PROBLEMA

- **Errore**: modifica di un bit (0 diventa 1 oppure 1 diventa 0)
- Posso avere uno o più errori

LA SOLUZIONE

Su una sequenza di bit, usare dei codici particolari che permettono:

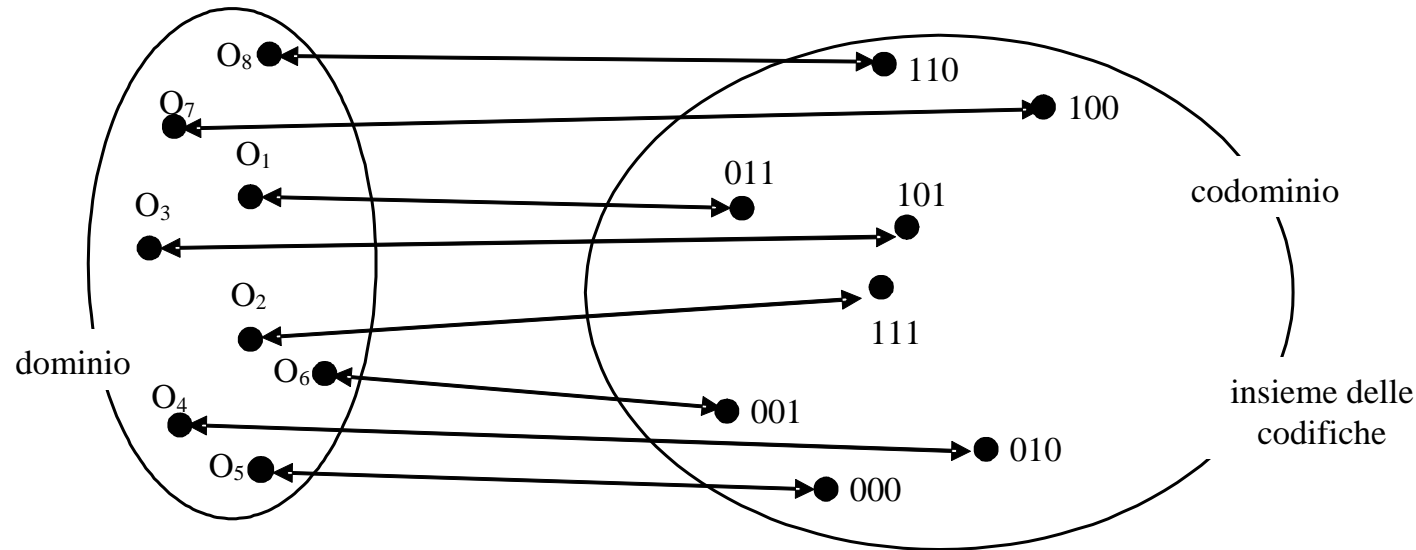
- Di rivelare la presenza di errori:

CODICI RILEVATORI

- Di correggere gli errori, ricostruendo la sequenza originaria:

CODICI CORRETTORI

COME?

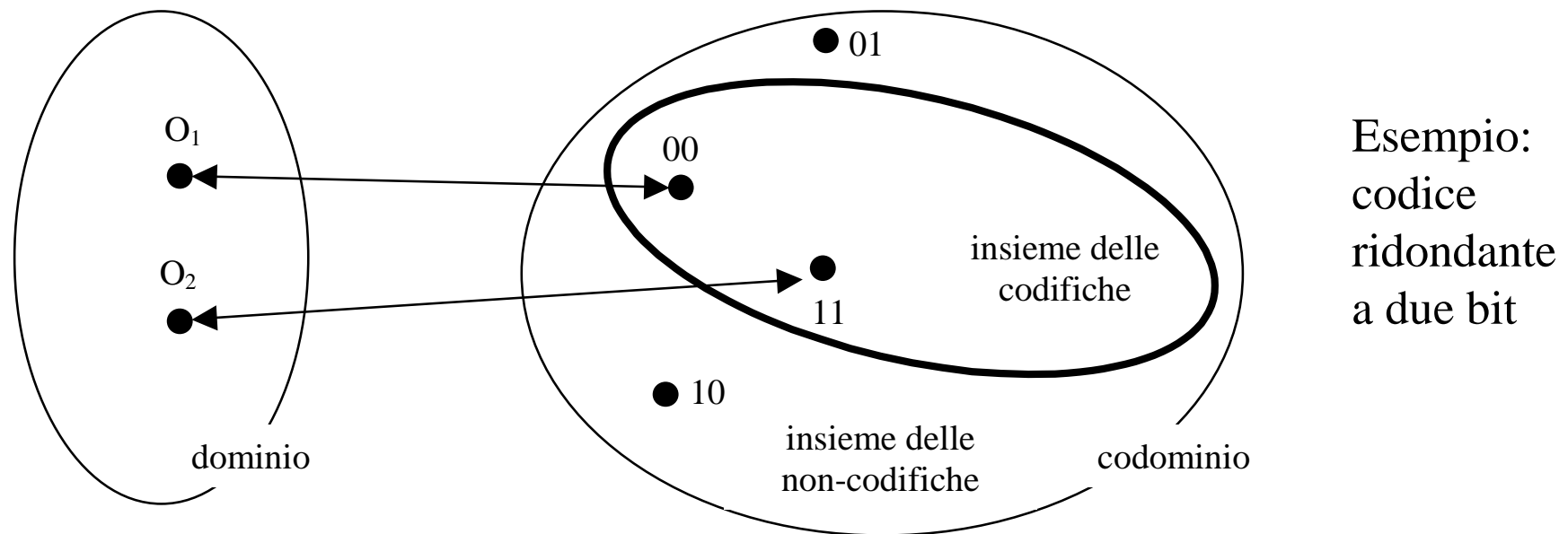


8 oggetti, 3 bit: il codice è *non ridondante*

➡ *Impossibile rilevare né correggere errori*

➡ Per rilevare o correggere errori dobbiamo introdurre della ridondanza

Codice *ridondante*: l'insieme delle codifiche è un sottoinsieme proprio del codominio \Rightarrow l'insieme delle non codifiche è non nullo



Per esempio:

se codifico O_1 e ho un errore (ad esempio sul primo bit)
ottengo 10 \Rightarrow rivelazione dell'errore

QUANTO?

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

QUANTO?

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

Esempio: un semplice codice per due entità, 5 bit

Atalanta



00000

Brescia



11111



Distanza tra le
codifiche: 5

QUANTO?

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

Esempio: un semplice codice per due entità, 5 bit

Atalanta



00000

Brescia



11111



Distanza tra le
codifiche: 5

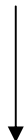
- Se ho un numero di errori $k \leq 4$: riesco a rilevarli
 $\Rightarrow d \geq k+1$

QUANTO?

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

Esempio: un semplice codice per due entità, 5 bit

Atalanta



00000

Brescia



11111

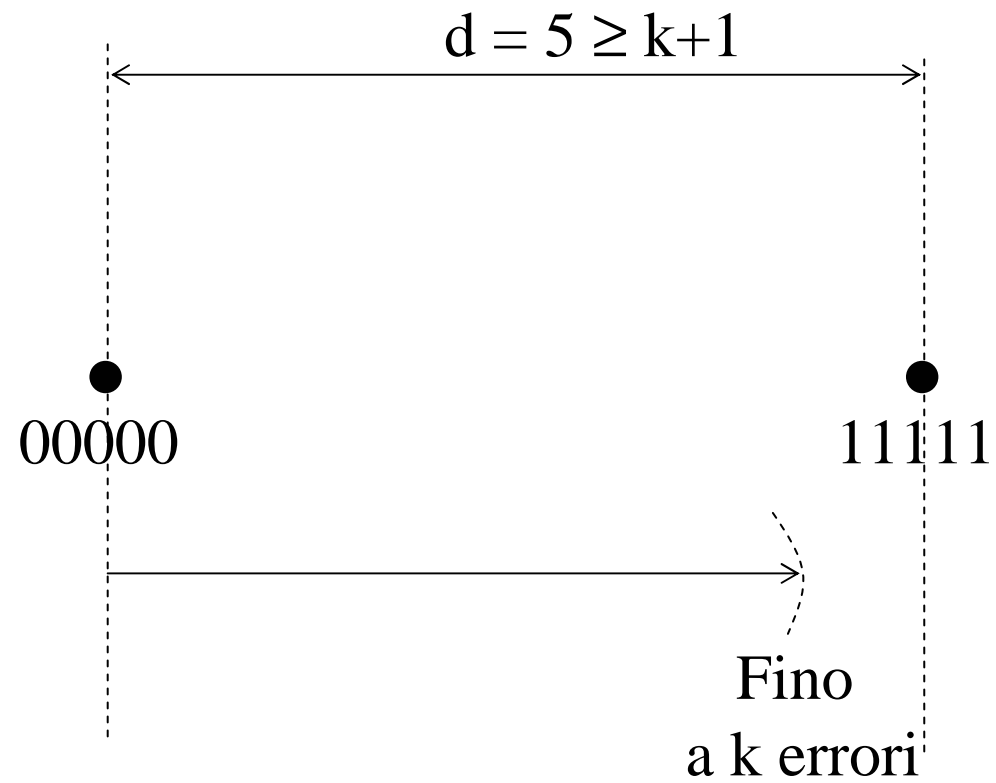


Distanza tra le
codifiche: 5

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

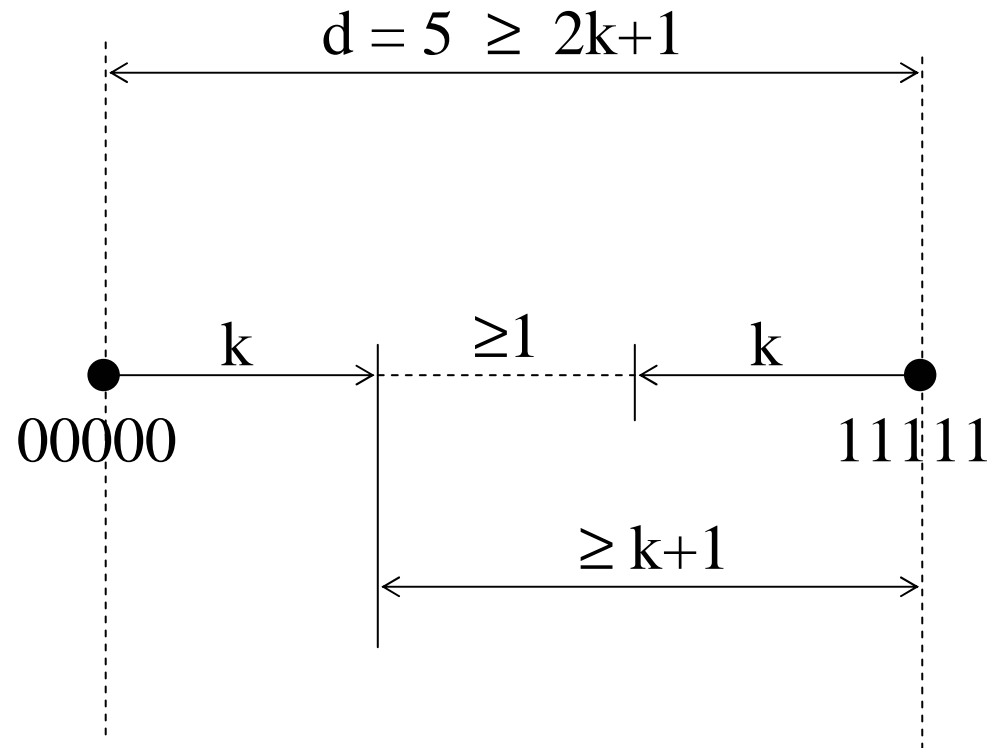
Rilevazione: l'idea

k=4



Correzione: l'idea

k=2



QUANTO IN GENERALE?

Distanza di un codice:

minimo valore delle distanze tra ogni coppia di codifiche

Rilevazione e correzione di errori

- Per rilevare 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 rilevatore: il **codice di parità**

- Dato un codice: si aggiunge un bit in modo che il numero di ‘uni’ sia pari [o dispari]

Esempio



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

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 rilevare 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 rilevati:

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

b) Errori corretti:

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

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 rilevare 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

- Cambiando modalità di codifica: ridurre il numero dei bit richiesti
- Vantaggi per *memorizzazione* e *trasmissione* (es: sequenze video)
- Concetti fondamentali validi per qualunque tecnica di compressione: data una sequenza di bit S
 - *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)