Programma del corso

- *Introduzione agli algoritmi*
- ***Rappresentazione delle Informazioni***
- *Architettura del calcolatore*
- *Reti di Calcolatori (Reti Locali, Internet)*
- *Elementi di Programmazione*

Numeri a precisione finita

- I numeri a precisione finita sono quelli rappresentati con un *numero finito di cifre*.
  - Fissate le caratteristiche del numero è determinato anche l'insieme di valori rappresentabili.

- Le operazioni con i numeri a precisione finita *causano errori* quando il loro risultato non appartiene all'insieme dei valori rappresentabili:
  - *Underflow*: si verifica quando il risultato dell'operazione è minore del più piccolo valore rappresentabile
  - *Overflow*: si verifica quando il risultato dell'operazione è maggiore del più grande valore rappresentabile
  - *Non appartenenza all'insieme*: si verifica quando il risultato dell'operazione, pur non essendo troppo grande o troppo piccolo, non appartiene all'insieme dei valori rappresentabili

Numeri a precisione finita

- Esempio: si considerino i numeri a tre cifre senza virgola e senza segno:

1	*5*	*9*

  - Non possono essere rappresentati:
    - Numeri superiori a 999
    - Numeri negativi
    - Frazioni e numeri irrazionali
  - Alcuni errori possibili in operazioni fra tali numeri:
    - 600+600 = 1200 ➔ *Overflow*
    - 300-600 = -300 ➔ *Underflow*
    - 007/002 = 3.5 ➔ *Non appartenenza all'insieme*

Numeri a precisione finita

- L'algebra dei dei numeri a precisione finita è diversa da quella convenzionale, poiché alcune delle proprietà non vengono rispettate.
  - A differenza dei numeri interi, i numeri a precisione finita *non rispettano la chiusura* rispetto alle operazioni di somma, sottrazione e prodotto.
  - La *proprietà associativa* [a + (b - c) = (a + b) – c] e la *proprietà distributiva* [a × (b - c) = a × b – a × c] non sono rispettate

- Esempi (numeri a precisione finita di 3 cifre senza virgola e senza segno):
  - Chiusura: 050×050=2500 (*Overflow*)
  - Prop. associativa: (400 + 300) – 500 = 200
    400 + (300 – 500) = *Underflow*
  - Prop. distributiva: 50 × (50 - 40) = 500
    50 × 50 – 50 × 40 = *Overflow*

I numeri reali e la loro rappresentazione

- Quando si parla di numeri "reali", nell'informatica, ci si riferisce sempre ad un piccolo sottoinsieme finito di numeri razionali.
- 2 possibili rappresentazioni:
  - in virgola fissa
  - In virgola mobile
- La prima rappresentazione potrebbe essere usata per applicazioni di tipo gestionale, dove l'ordine di grandezza dei numeri che compaiono non è mai troppo diverso
- La seconda invece consente di gestire numeri il cui ordine di grandezza è profondamente differente

Esempio:
Massa dell'elettrone: 0.00000000000000000000000000000091 Kg
La massa della terra è: 597360000000000000000 Kg

Rappresentazioni in virgola fissa

Una prima possibile rappresentazione dei numeri razionali è la rappresentazione in virgola fissa che riserva un numero fisso di bit per parte intera e parte frazionaria.

Per semplicità nei prossimi esempi consideriamo solo numeri positivi

Es.

Avendo **5 bit** a disposizione potremmo utilizzare **3 bit** per la parte intera e **2 bit** per quella frazionaria.

Numeri floating point

- Molte applicazioni richiedono il trattamento di valori razionali o reali
  - 1/3 = 0.333333… π = 3.14159265…
  - Non rappresentabili con un numero finito di bit
  - Per numeri molto grandi spesso interessano solo le cifre *più significative*
- Si adotta una notazione in cui la gamma dei valori esprimibili è indipendente dal numero di cifre significative. Questo sistema è detto *floating-point*.

frazione o mantissa

$$n = f \times 10^{e}$$

esponente

La precisione è determinata dalla mantissa *f*, mentre la gamma dei valori è determinato dall'esponente *e*.

Esempio: *Valori floating point corrispondenti alla mantissa f=0.241, al variare del numero delle cifre significative e dell'esponente e.*

c.s. \ e	-3	-2	-1	0	1	2	3
1	0.0002	0.002	0.02	0.2	2	20	200
2	0.00024	0.0024	0.024	0.24	2.4	24	240
3	0.000241	0.00241	0.0241	0.241	2.41	24.1	241

Rappresentazioni in virgola mobile

Ad esempio:

In **base 10** dato il numero reale 474.35 può avere varie rappresentazioni in virgola mobile data dalla coppia (M,E):

$(0.47435, +3)$, cioè $0.47435\ x\ 10^{3}$

$(0.047435, +4)$, cioè $0.047435\ x\ 10^{4}$

$(47435.0, -2)$, cioè $47435\ x\ 10^{-2}$

Rappresentazioni in virgola mobile

Ad esempio:

In **base 2,** dato il numero binario 101.011 può avere varie rappresentazioni in virgola mobile data dalla coppia (M,E):

$(1.01011, +2)$, cioè $1.01011\ x\ 2^2$

$(0.0101011, +4)$, cioè $0.0101011x\ 2^4$

$(101011.0, -3)$, cioè $101011.0x\ 2^{-3}$

Numeri floating point

Con lo stesso metodo possiamo rappresentare numeri molto grandi. Ad esempio, con 8 bit:

4 bit di mantissa: 1111 = 15

4 bit di esponente: 1111 = 15

1111 1111 = $15 * 2^{15} = 491520$

Mentre, con la notazione classica, con 8 bit rappresentiamo al massimo il numero 255 !

Numeri floating point

Ma allora, perchè non usare sempre la notazione floating point?

Perchè si perde in precisione

Esempio: 5 cifre (decimali) : 4 per la mantissa, 1 per l'esponente. Rappresentare

312,45

<3124; -1> = [312,4 ... 312,5]???

Non posso rappresentare numeri qua dentro!

Numeri floating point

- Non tutti i numeri reali appartenenti alle aree rappresentabili possono essere espressi correttamente tramite un numero floating-point.
  - Esempio: *Con numeri floating-point con tre cifre decimali con segno per la mantissa e due cifre decimali con segno per l'esponente non è possibile rappresentare* 10/3=3.3333333...

$$0.333 \times 10^1 < 3.\overline{3} < 0.334 \times 10^1$$

A differenza dei numeri reali, la *densità* dei numeri floating-point *non è infinita* ➔ ***Errori di arrotondamento***

- Quando il risultato v non si può esprimere nella rappresentazione numerica adottata, si utilizza il numero più vicino rappresentabile ($v_1 < v < v_2$).

Codifica di immagini

Codifica di immagini

- Un’immagine è un insieme continuo di informazioni
  - A differenza delle cifre e dei caratteri alfanumerici, per le immagini non esiste un'unità minima di riferimento
- Problema: rendere **digitale** una informazione prettamente **analogica**

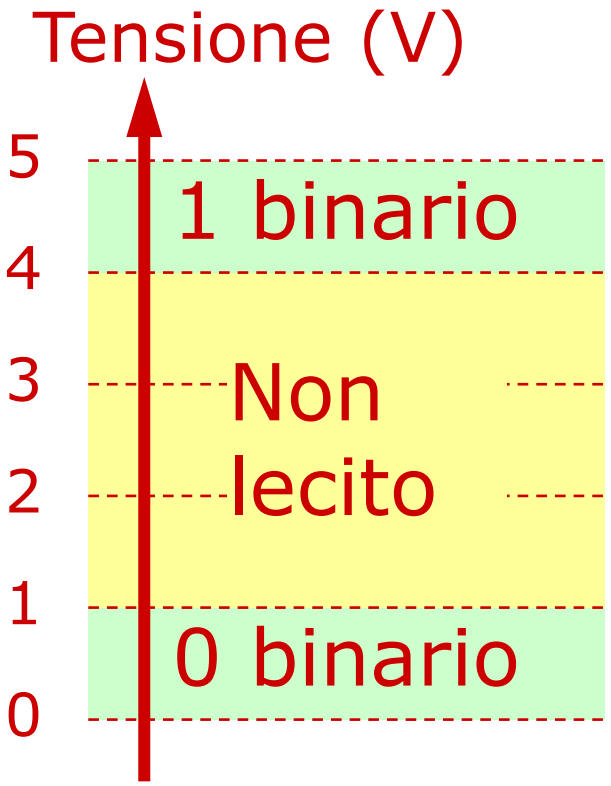
Analogico e digitale

- Meta-informazione esplicita nel supporto
- Codifica ANALOGICA

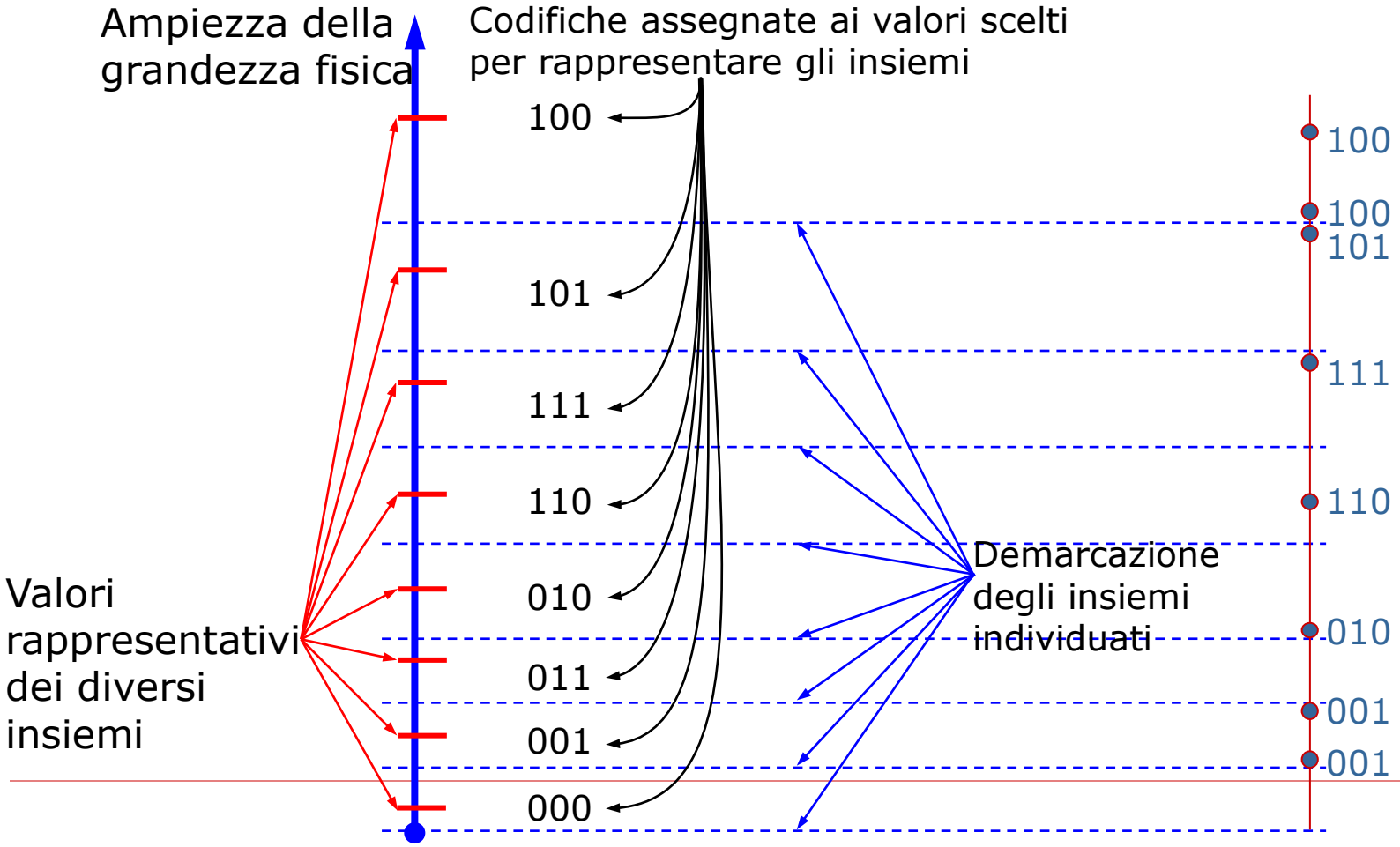
- Meta-informazione implicita nella codifica
- Codifica DIGITALE

Il successo del digitale

- Rumore: effetto dell'ambiente sul supporto.
- Quanto un supporto è "immune" al rumore?
  - Codifica analogica: ogni configurazione è lecita dal punto di vista informazionale e quindi risulta impossibile distinguere il rumore dal segnale.
  - Codifica digitale: un valore binario è associato a un insieme di configurazioni valide quindi si può
    - riconoscere il rumore che porta in configurazioni non lecite
    - trascurare il rumore che non fa uscire il segnale dall'insieme associato alla stessa configurazione

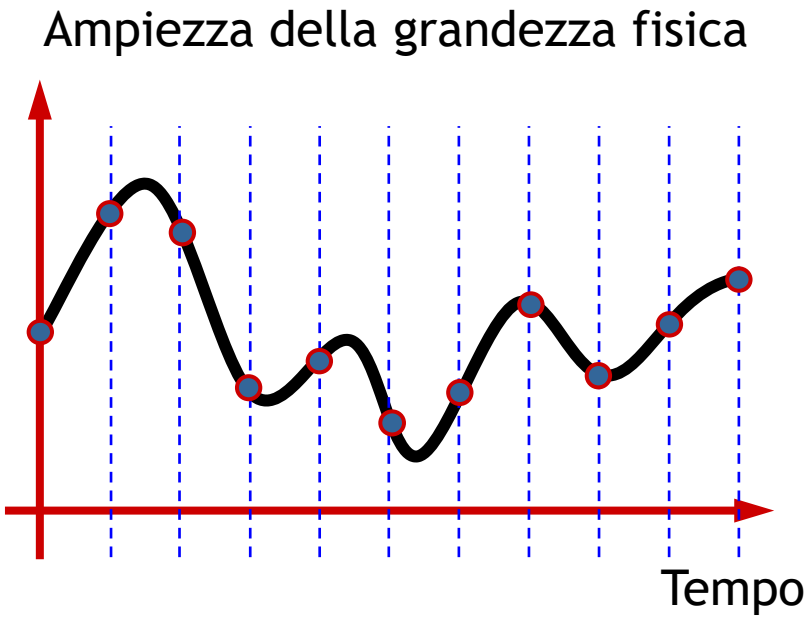


Da analogico a digitale: la quantizzazione

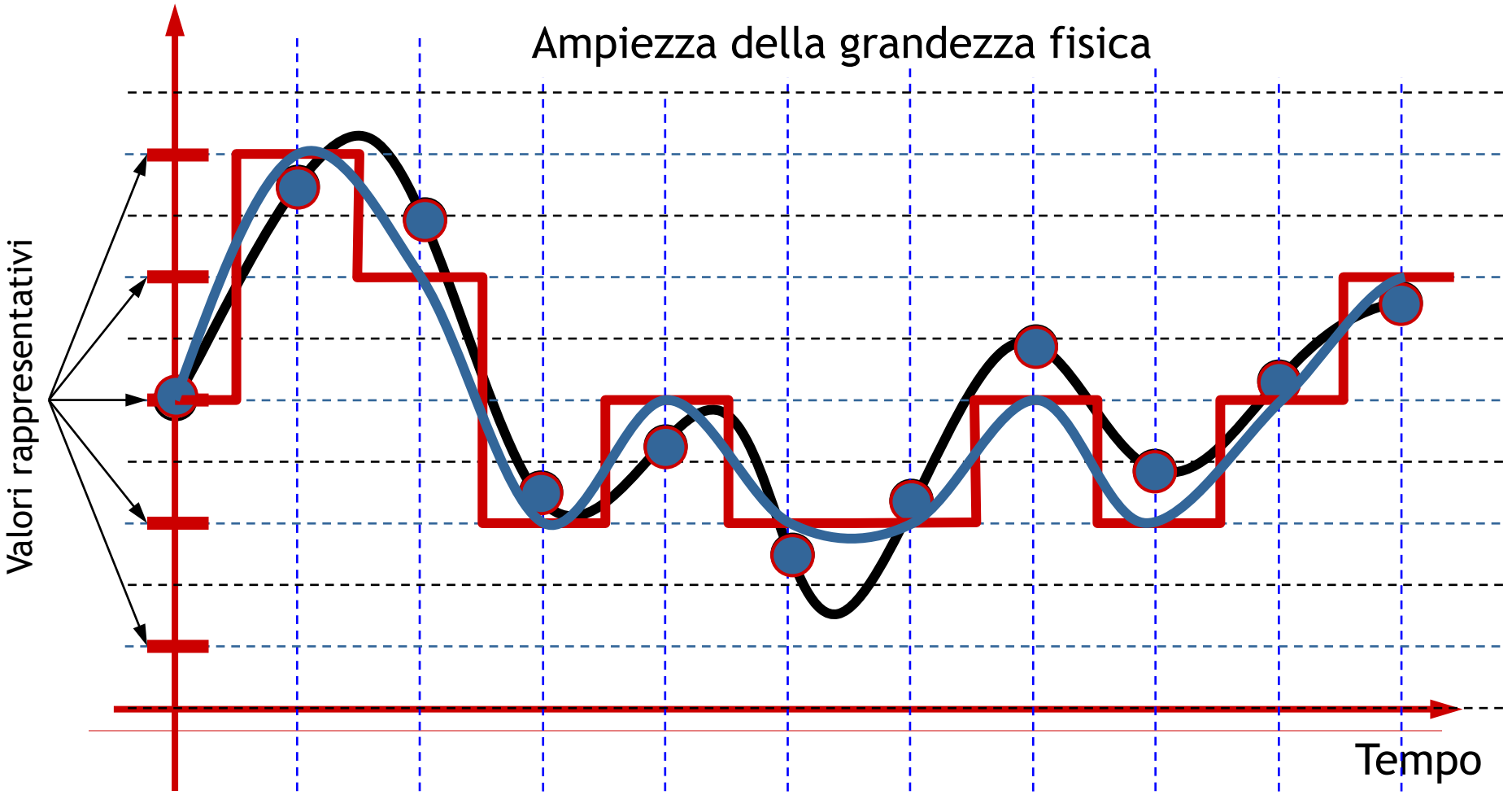


Da analogico a digitale: frequenza di campionamento

- La grandezza varia nel tempo e non può essere rappresentata da un solo valore.
- I valori di riferimento debbono essere rilevati in diversi istanti di tempo (frequenza di campionamento).
- La quantizzazione deve poi essere ripetuta per ogni valore campionato.



Campionamento e quantizzazione



Codifica dei suoni

- Rappresentazione tanto più precisa tanto più
  - Frequente è la campionatura
  - Maggiore il numero di bit per codificare l'informazione
- Esempi
  - Cassetta musicale: 22 KHz, 9/10 bit per campione
  - Audio CD: 44,1 KHz, 16 bit per campione
  - DVD: 48 KHz, 16 bit per campione
  - Schede audio PC: 10/40 KHz, 16 bit per campione

Tecniche di compressione

- MP3 (MPEG 1 Layer 3) utilizza una tecnica di compressione con perdita dei dati
  - Il formato MP3: usa dei criteri psico-acustici umani:
    - considera solo i suoni che il nostro cervello è in grado di percepire (sotto frequenza 20000Hz e non nascosti da suoni di intensità maggiore)
  - Risparmio di 1/10 rispetto a formato .wav (bitrate 128 Kbit/s vs bitrate 1411 Kbit/s) dove bitrate = num di bit necessari per codificare 1 secondo di audio

Codifica dei suoni

Alcuni formati:

.mov

.wav

.mpeg

.avi

.midi - usato per l'elaborazione della musica al computer

Campionamento a 16 bit

	Free	Premium
Lettore Web	AAC 128 kbit/s	AAC 256 kbit/s
Desktop, dispositivi mobili e tablet	• **Automatica:** dipende dalla connessione di rete. • **Bassa:** equivalente approssimativamente a 24 kbit/s. • **Normale:** equivalente approssimativamente a 96 kbit/s. • **Alta:** equivalente approssimativamente a 160 kbit/s.	• **Automatica:** dipende dalla connessione di rete. • **Bassa:** equivalente approssimativamente a 24 kbit/s. • **Normale:** equivalente approssimativamente a 96 kbit/s. • **Alta:** equivalente approssimativamente a 160 kbit/s. • **Molto alta (solo Premium):** equivalente approssimativamente a 320 kbit/s.

Codifica di immagini

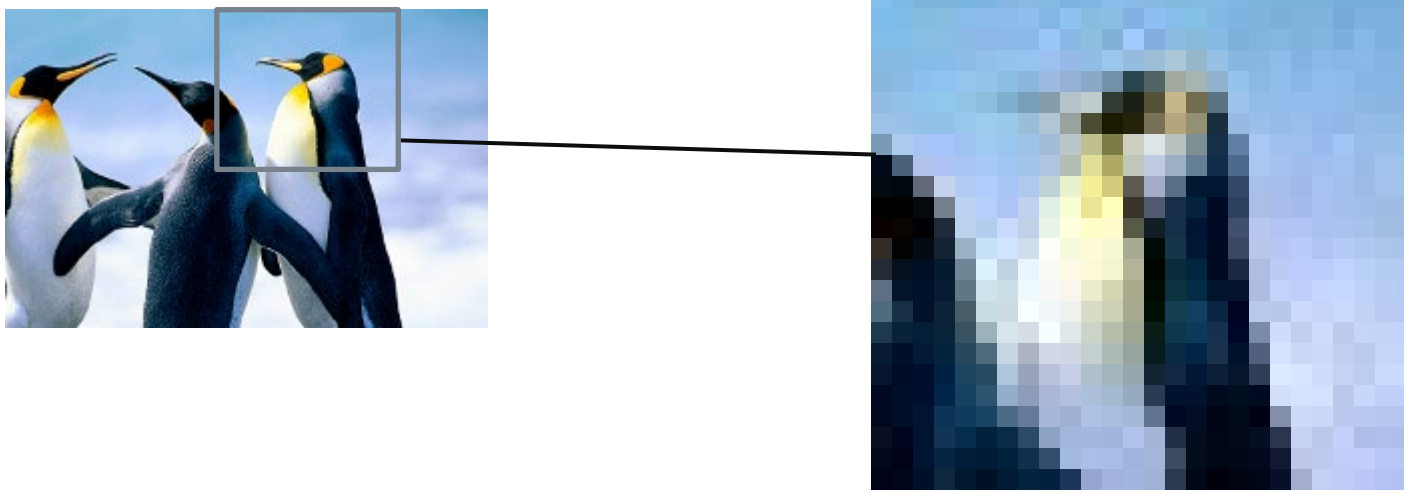
- Esistono numerose tecniche per la memorizzazione digitale e l'elaborazione di un'immagine
  - Una prevede la scomposizione dell'immagine in una *griglia* di tanti elementi (**punti**) che sono l'unità *minima* di memorizzazione - **Formato Raster/Bitmap**
  - La seconda strada prevede la presenza di strutture elementari di natura più complessa, quali *linee*, *circonferenze*, *archi*, etc. - **Formato Vettoriale**

Immagini Digitali

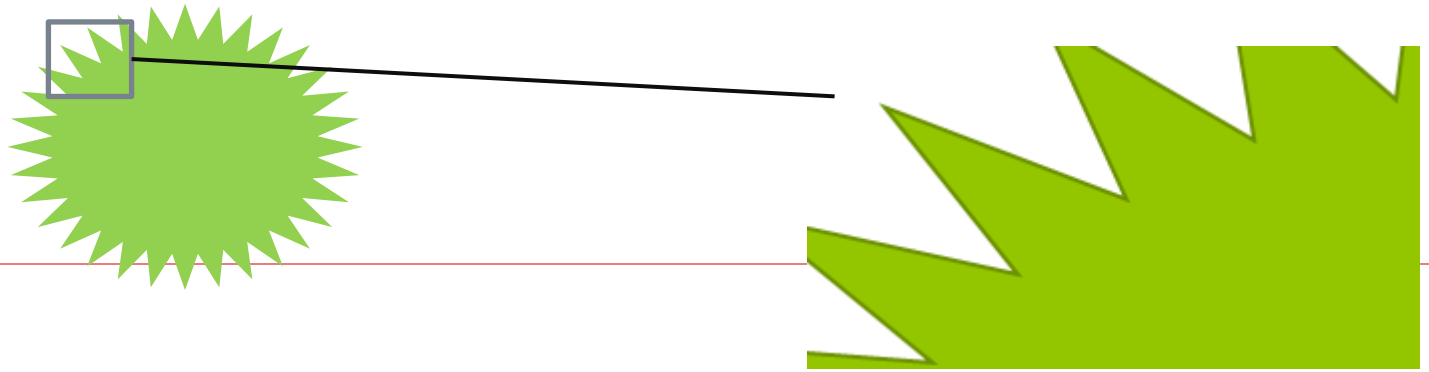
Tipo	*Definizione*	Proprietà
Raster o Bitmap	Mappata all'interno di una griglia, come un grande mosaico.	Rappresentazione più ***semplice*** (richiesta poca elaborazione) Spazio maggiore per essere memorizzate.
Vettoriale	Basate su forme e colori generate tramite formule matematiche	Ingrandimento teoricamente infinito. Rappresentazione più complessa

Immagini Digitali: Ingrandimento

Raster o **Bitmap**



Vettoriale

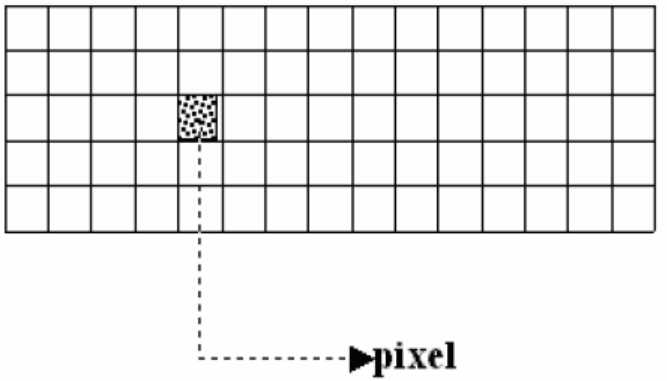


Codifica delle immagini B/N

- Dividere l'immagine in una griglia a righe orizzontali e verticali
- Ogni quadratino della griglia è un **pixel** (**picture element**)
- Codificare ogni pixel con:
  - 0 se il pixel è bianco
  - 1 se il pixel è nero
- Convenire un ordinamento per i bit usati nella codifica

Rappresentazione di immagini

- Codifica binaria delle immagini
  - L'immagine è suddivisa in "punti" o picture element (*pixel*).
    - Di ogni pixel viene rappresentato il colore o livello di grigio, espresso da un codice numerico intero.
    - L'immagine diviene quindi una lunga sequenza di bit (*digitalizzazione*).

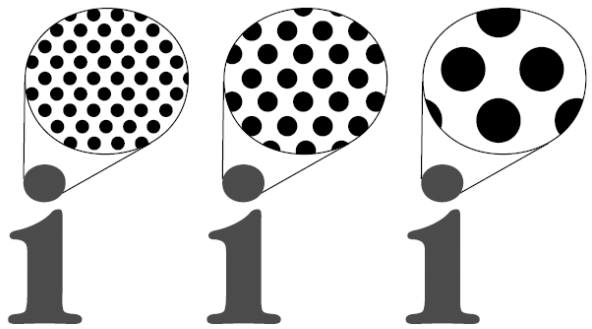


  - La *risoluzione* di un'immagine è determinata dal numero di punti per pollice (dot per inch o *dpi*) e dal numero di colori o livelli di grigio (fissato dal numero di bit con cui si codifica tale caratteristica).
    - Per ricostruire l'immagine a partire dalla sequenza di bit bisogna conoscerne la base, l'altezza (espresse in pixel) e il numero di bit per pixel.

Pixels

- Pixel = picture element, è il più piccolo elemento di una griglia in cui è diviso lo schermo. A ogni pixel si assegna un indirizzo in memoria, così che il computer può deciderne cololore e luminosità.
- Cella = unità minima nella quale è divisa la pagina stampata. Le sfumature di colore dipendono dalla tonalità preponderante nei puntini all'interno della cella.

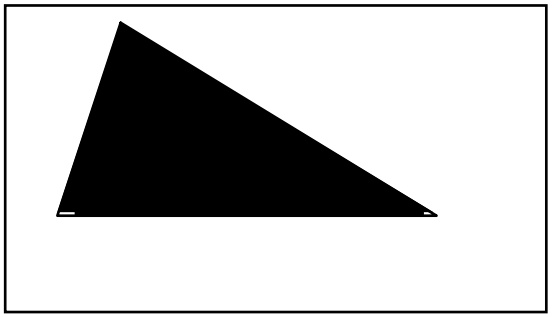
- Risoluzione = qualità di un'imagine:
  - Schermi -> numero di pixel (es: 640x480, 1024x768)
  - Stampanti -> dpi (dot per inch, punti per pollice, ovvero numero di punti su una linea lunga 2,54 cm)



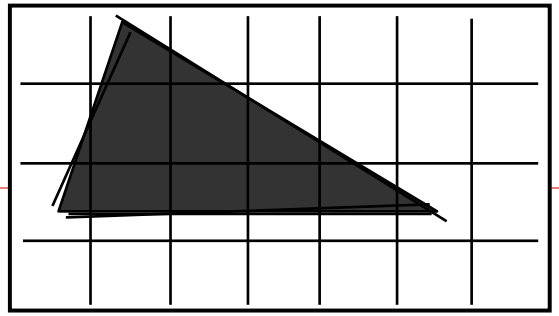
Diversi gradi di risoluzione

Codifica delle immagini B/N

- Consideriamo un'immagine in bianco e nero, senza ombreggiature o livelli di chiaroscuro



- Suddividiamo l’immagine mediante una griglia formata da righe orizzontali e verticali a distanza costante



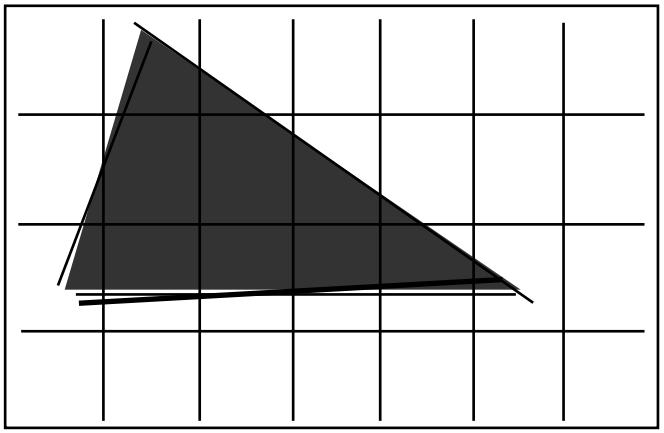
Codifica delle immagini B/N

- Ogni quadratino derivante da tale suddivisione prende il nome di **pixel** (picture element) e può essere codificato in binario secondo la seguente convenzione:
  - il simbolo "**0**" viene utilizzato per la codifica di un pixel corrispondente ad un quadratino bianco (in cui il bianco è predominante)
  - il simbolo "**1**" viene utilizzato per la codifica di un pixel corrispondente ad un quadratino nero (in cui il nero è predominante)

Codifica delle immagini B/N

Poiché una sequenza di bit è lineare, si deve definire una convenzione per **ordinare** i pixel della griglia

Hp: assumiamo che i pixel siano ordinati dall'alto verso il basso e da sinistra verso destra



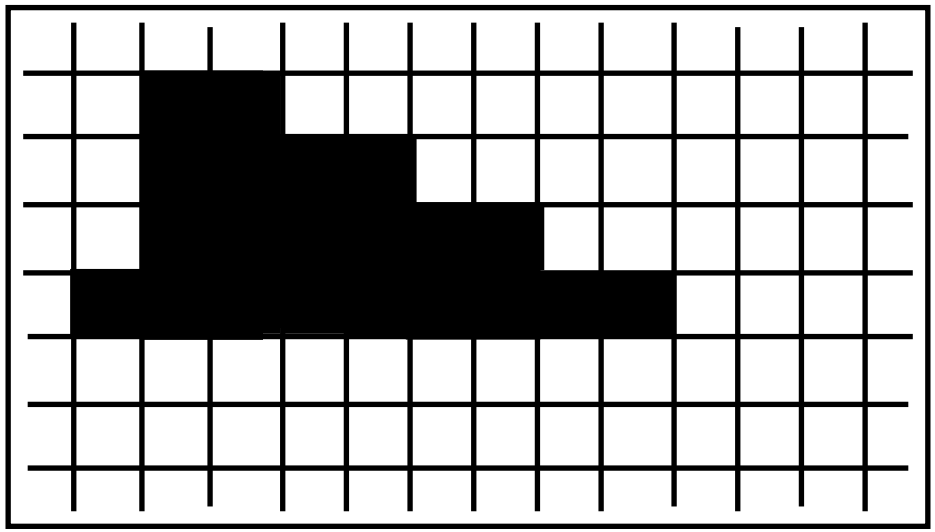
0_{1}	1_{2}	0_{3}	0_{4}	0_{5}	0_{6}	0_{7}
0_{8}	1_{9}	1_{10}	0_{11}	0_{12}	0_{13}	0_{14}
0_{15}	1_{16}	1_{17}	1_{18}	1_{19}	0_{20}	0_{21}
0_{22}	0_{23}	0_{24}	0_{25}	0_{26}	0_{27}	0_{28}

La rappresentazione della figura è data dalla stringa binaria

0100000 0110000 0111100 0000000

Codifica delle immagini B/N

- Non sempre il contorno della figura coincide con le linee della griglia
  - nella codifica si ottiene un'approssimazione della figura originaria
- La rappresentazione sarà più fedele all'aumentare del numero di pixel
  - ossia al diminuire delle dimensioni dei quadratini della griglia in cui è suddivisa l'immagine



Codifica delle immagini B/N

Quindi: le immagini sono rappresentate con un certo livello di approssimazione, o meglio, di **risoluzione**, ossia il numero di pixel usati per riprodurre l'immagine.

Risoluzioni tipiche

- 640 x 480 pixel; 800 x 600 pixel
- 1024 x 768 pixel; 1280 x 1024 pixel

Immagini in toni di grigio

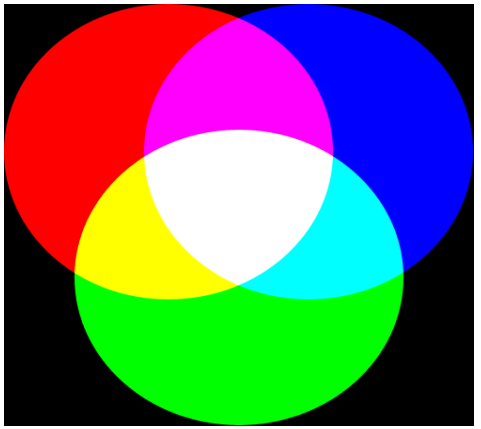
- Le immagini in bianco e nero hanno delle sfumature, o **livelli di intensità di grigio**

- Per codificare immagini con sfumature:
  - si fissa un insieme di livelli (*toni*) di grigio, cui si assegna convenzionalmente una rappresentazione binaria
  - per ogni pixel si stabilisce il livello medio di grigio e si memorizza la codifica corrispondente a tale livello

- Per memorizzare un pixel non è più sufficiente 1 bit.
  - con **4** bit si possono rappresentare $\mathbf{2^4=16}$ livelli di grigio
  - con **8** bit ne possiamo distinguere $\mathbf{2^8=256}$,
  - con **K** bit ne possiamo distinguere $\mathbf{2^K}$

Immagini a colori

- Analogamente possono essere codificate le immagini a colori:
  - bisogna definire un insieme di sfumature di colore differenti e rappresentarle mediante una opportuna sequenza di bit
- Nella codifica **RGB** si utilizzano tre colori
  - **rosso** (Red), **verde** (Green) e **blu** (Blue)
- Ad ogni colore si associa un certo numero di sfumature codificate su N bit
  (2^N possibili sfumature)



- Esempio
  - con 2 bit per colore si ottengono 4 sfumature per colore
  - con 8 bit per colore si ottengono 256 sfumature per colore e 256^3(16 milioni) possibili colori

Immagini a colori

- La qualità dell'immagine dipende
  - dal numero di punti in cui viene suddivisa (*risoluzione*)
  - dai toni di colore permessi dalla codifica;

Bitmap (o immagine raster)

- La rappresentazione di un'immagine mediante la codifica a pixel viene chiamata **bitmap**
- Il numero di byte richiesti per memorizzare un bitmap dipende dalla risoluzione e dal numero di colori
- Esempio
  - se la risoluzione è 640x480 con 256 colori occorrono 2.457.600 bit = 307200 byte = circa 300 KB

Bitmap

- I formati bitmap più conosciuti sono
  - **BITMAP** (.bmp),
  - **GIF** (.gif),
  - **JPEG** (.jpg)
  - **TIFF** (.tiff)

  } compressi

- In tali formati si utilizzano metodi di ***compressione*** per ridurre lo spazio di memorizzazione
  - Aree dello stesso colore si rappresentano in modo "abbreviato".

- E' in genere possibile passare da un formato ad un altro

Compressione dei dati

- Lossless
  - Senza perdita di informazione
  - Programmi, documenti

- Lossy
  - Con perdita di informazione
  - Rapporto di compressione variabile dall'utente
  - Immagini: GIF, JPEG (elimina lievi cambiamenti di colore)
  - Animazioni: MPEG (memorizza solo differenze tra fotogrammi)
  - Audio: MP3 (elimina suoni a basso volume sovrapposti con suoni ad alto volume)

Tecniche di compressione

- Le immagini possono richiedere molto spazio per la loro memorizzazione
- Esempi di tecniche di compressione
  - 000000000011 → 10 volte 0, 2 volte 1
  - Memorizzazione non di tutti i bit o fotogrammi (riduzione di fedeltà rispetto all'originale ma spesso non è percepibile dall'occhio umano)
  - Es. MPEG (filmati): un fotogramma ogni 12

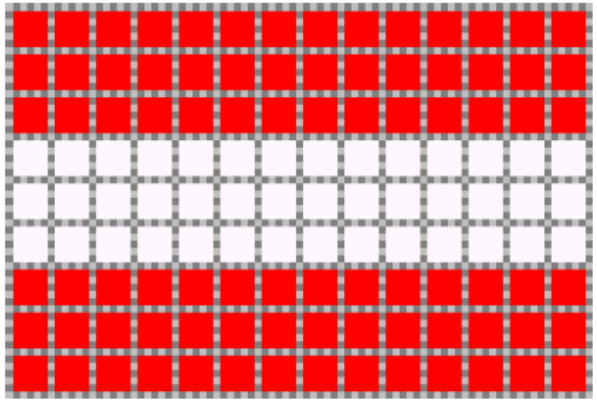
Un semplice esempio con dizionario

- Compressione lossless con tecnica basata su un dizionario
  - Testo di esempio:
    "I re di Francia della dinastia Carolingia sono: Carlo II, Luigi II di Francia, Luigi III di Francia, Carlomanno di Francia, Carlo III detto il grosso, Odo, Carlo III detto il semplice, Roberto I di Francia, Rodolfo Duca di Borgogna, Luigi IV di Francia, Lotario di Francia, Luigi V di Francia" (lunghezza: 292 caratteri)
- Analisi delle regolarità presenti nel testo:
  - Si individuano le sequenze ripetute (parole) contando le ripetizioni e si compila il dizionario (vedere tabella)
  - Si assegna un simbolo che la sostituisce ad ogni parola
  - Il testo diventa:
    "I re 1 2 della 1nastia Carolingia sono: 5 3, 4 3 1 2, 4 3I 1 2, 5manno 1 2, 5 3I 6 7 grosso, Odo, 5 3I 6 7 semplice, Roberto I 1 2, Rodolfo Duca 1 Borgogna, 4 IV 1 2, Lotario 1 2, 4 V 1 2" (lunghezza: 187 caratteri + 35 caratteri per il dizionario = 222 caratteri - 76% della lunghezza originaria)
  - Un testo più lungo permette una efficienza maggiore!

Indice	Parola	N
1	di	10
2	Francia	8
3	II	5
4	Luigi	4
5	Carlo	4
6	detto	2
7	il	2

Rappresentazione di immagini (2)

- Esempio
  - Immagine di 14x9 pixel a 8 bit per pixel (256 colori)
  - Si supponga 3=rosso e 7=bianco



3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3
7	7	7	7	7	7	7	7	7	7	7	7	7	7
7	7	7	7	7	7	7	7	7	7	7	7	7	7
7	7	7	7	7	7	7	7	7	7	7	7	7	7
3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3

Dimensione: 126 byte

- Un modo alternativo per rappresentare la stessa immagine:

42	3	42	7	42	3

Dimensione: 6 byte

Qualità 90/100

800x600

16,8mln colori 24 bit

Bitmap: 1440000 byte

JPEG: 258971 byte

Oualità 25/100

Qualità 10/100

Codifica vettoriale delle immagini

- Si utilizza quando le immagini da memorizzare hanno caratteristiche geometriche ben definite
- Il disegno da memorizzare può essere facilmente *scomposto in elementi base* come una linea o un arco di circonferenza
- La memorizzazione dell'intera immagine avviene tramite la codifica di ogni singola parte

Codifica vettoriale delle immagini

- Richiede poco spazio
- Per definire un segmento basteranno le coordinate dei due estremi (Linea dal punto <10;12> a <20; 30>)
- Il formato più diffuso è il **PostScript** (ps, eps)
  - usato anche per la stampa dei testi
- Altri formati: wmf, cdr, ai (es:**CorelDraw, Illustrator, Inkscape**)