Sistemi Multimediali
I formati dei media

Ombretta Gaggi
Università di Padova

Multimedia per …

testo, immagini, audio, video per…

…il Web?

…Internet?

…i sistemi di rete?

…i sistemi distribuiti?

…tutti i casi elencati?

quali sono le differenze?

WWW vs. Internet

Il World Wide Web è un'applicazione distribuita (un contenitore di applicazioni…) che usa Internet come infrastruttura tecnologica

- Internet fornisce servizi di comunicazione, protocolli, sistemi di archiviazione remota, etc.
- WWW fornisce l'interfaccia utente, un ambiente per le applicazioni distribuite e le applicazioni

I sistemi multimediali coprono entrambi i mondi

- ma ogni mondo ha il proprio spazio di utilizzo
- a volte i confini sono poco definiti

WWW e tecnologie multimediali

Le tecnologie multimediali sono ampiamente indipendenti dal Web

- rappresentazione dell'informazione
- compressione dell'informazione
- trasmissione dell'informazione

Le applicazioni Web non sono le più critiche, né quelle che richiedono maggiori risorse

- es. video on demand
- es. comunicazione multicanale
- l'evoluzione del Web sta comunque imponendo nuovi obiettivi e quindi nuovi standard di qualità

Argomenti

Classificazione dei media

Proprietà dei media

Compressione delle informazioni

Immagini
- generalità, GIF, JPEG, PNG

Audio
- generalità, psicoacustica, MP3, MIDI

Video
- generalità, MPEG, H.261, H.263, DivX, XviD

Classificazione dei media

In base al contenuto dei media
- testo, immagine, audio, video
- il testo rappresenta un caso banale

In base alla dinamica dei media
- diffusione e uso dei media
- protocolli e tecnologie di rete

Classificazione dei media: dinamica

Media statici
- l'informazione non cambia nel tempo

Media dinamici
- l'informazione cambia nel tempo

Media temporizzati

- l'informazione cambia nel tempo ed è soggetta a vincoli temporali

Classificazione dei media:tipo

Immagine
- codifica, compressione, qualità
- pittorica (pixel) vs. geometrica (disegno vettoriale)

Audio
- codifica, compressione, qualità
- voce vs. musica vs. generico

Video
- codifica, compressione, qualità
- animazione vs. filmato

dimensione vs. tempo vs. qualità

Obiettivi della codifica di dati multimediali

- ✓ Ridurre la dimensione di una rappresentazione persistente dei dati
- ✓ Ottimizzare il tempo di trasmissione rispetto al tempo di riproduzione per dati continui
- ✓ Consentire la decodifica veloce di dati codificati su architetture convenzionali e non
- ✓ Controllare la qualità dei dati decodificati
- ✓ Fornire supporto per la sostituzione di dati mancanti

Media statici (1)

L'utente legge le informazioni con il proprio ritmo

La diffusione dei dati non impone scadenze temporali (entro limiti ragionevoli…)

L'informazione diffusa è persistente

L'informazione può essere letta più volte

Es. testo,immagini

Media statici (2)

La dimensione dei dati non costituisce problema

- maggiore dimensione = maggior tempo
- compressione / decompressione

La diffusione dell'informazione e il suo uso avvengono in momenti separati

- download / display
- archiviazione sul disco locale
- caching

Media dinamici (1)

L'utente deve seguire la dinamica dell'informazione in tempo reale

La diffusione può essere ritardata ma il significato dell'informazione può cambiare (es. giochi)

L'informazione non è persistente (a volte può essere riprodotta più volte)

Es. presentazioni animate, introduzioni, Flash / Shockwave

Media dinamici (2)

La dimensione dell'informazione è spesso grande, e deve essere fruita in un tempo predefinito ragionevole

Diffusione e uso dell'informazione sono contemporanee

- download vs. streaming
- soluzioni proprietarie
- controllo dell'interazione utente

Media temporizzati (1)

L'utente deve accedere all'informazione secondo tempi propri dell'informazione stessa

La diffusione deve rispettare l'isocronismo dei media

L'informazione diffusa non è persistente

I dati non possono essere fruiti due volte

Devono essere riprodotti ad una certa frequenza costante per poter essere compresi

Es. audio e video real-time

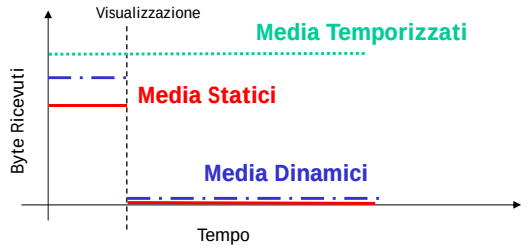
Media temporizzati (2)

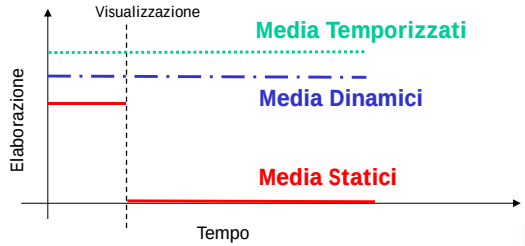
La dimensione dell'informazione è grande e proporzionale al tempo di riproduzione

Il ritmo di diffusione dei dati deve essere controllato

- real-time
- streaming
- ampiezza di banda

Media statici vs dinamici vs temporizzati





La compressione dei dati

La dimensione dei dati multimediali è troppo grande perché si possano trasmettere o archiviare informazioni in modo efficiente senza ricorrere a tecniche di compressione

- audio di qualità CD (2 canali, 16 bit a 44.1 kHz): 172 Kbyte/s
- video di qualità VHS (382*288*16 bit a 25 frame/s): 5.2 Mbyte/s
- video di qualità NTSC (640*480*24 bit a 30 frame/s): 26.4 Mbyte/s
- video Full HD (1080*1920*24 bit a 25 frame/s): 148 Mbyte/s

La compressione dei dati ha lo scopo di ridurre lo spazio necessario alla loro memorizzazione riducendo anche il tempo necessario per la trasmissione

- il dato deve essere decompresso per essere utilizzato
- la compressione può essere eseguita in tempo reale o in tempo differito
- la decompressione deve essere eseguita in tempo reale
- il dato deve essere fruibile ma non necessariamente identico all'originale (solo per dati multimediali)

La compressione dei dati

La riduzione dello spazio occupato può avvenire attraverso:

- la riduzione del numero di bit necessari per codificare una singola informazione (compressione *entropica*)
- la riduzione del numero di informazioni da memorizzare o trasmettere (compressione differenziale, compressione semantica)

La compressione può conservare integralmente o no il contenuto del dato originale

- compressione senza perdita di informazione (*lossless*, reversibile): sfrutta le ridondanze nella codifica del dato
- compressione con perdita di informazione (*lossy*, irreversibile): sfrutta le ridondanze nell'utilizzo (percezione) del dato

Il processo di compressione

Il processo di compressione procede attraverso tre fasi:

- *trasformazione*: i dati che costituiscono l'informazione (es. i punti di un'immagine) sono trasformati in un dominio che richiede (può richiedere) meno bit per la rappresentazione dei valori
- *quantizzazione*: i valori così ottenuti sono (possono essere) raggruppati in un numero di classi minore e con distribuzione più uniforme
- *codifica*: l'informazione viene codificata in termini delle classi e dei valori ottenuti, insieme alla tabella di codifica

Tecniche di compressione lossless

Run-length encoding (RLE)

- codifica sequenze di valori uguali premettendo un indicatore di ripetizioni al valore codificato

Codifica di Huffman

- assegna un numero inferiore di bit alle sequenze più probabili attraverso un vettore di codifica

Compressione Lempel-Ziv-Welch (LZW)

- costruisce dinamicamente una tabella di codifica con numero variabile di bit sulla base delle sequenze incontrate

Codifica differenziale

- ogni dato è rappresentato come differenza rispetto al dato precedente (con risoluzione lineare o non lineare)

Tecniche di compressione lossy

Compressione JPEG (per le immagini)

- applica all'immagine una trasformata nel dominio delle frequenze (Discrete Cosine Transform) che permette di sopprimere dettagli irrilevanti riducendo il numero di bit necessari per la codifica
- consente anche compressione *lossless* con tecniche predittive

Compressione MPEG (per i video)

- codifica parte dei frame come differenze rispetto ai valori previsti in base ad una interpolazione

Compressione MP3 (per l'audio)

- si basa alle proprietà psicoacustiche dell'udito umano per sopprimere le informazioni inutili

Run Length Encoding

Se ci sono molti valori consecutivi ripetuti c'è molta ridondanza che può essere eliminata

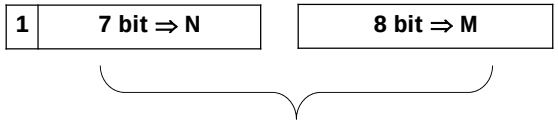
- lunghe sequenze dello stesso valore possono essere sostituite da un simbolo di ripetizione, dal valore e dal numero di ripetizioni
- la compressione è efficace per sequenze di almeno tre valori uguali
- è adatta alla rappresentazione di immagini con poco dettaglio (es. uno sfondo uniforme)
- è implementata nel formato bit-map BMP

Esempio

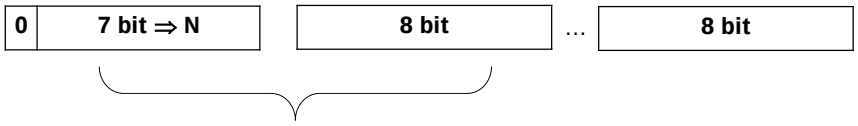
- BGFDDDDDDDDDIJUPPPPPHYTGBUYYYYINNNNNNGHHHHHHHKPPPPPPP
- BGF@9DIJU@5PHYTGBU@4YI@5NG@7HK@7P

Pacchetti RLE

RUN LENGHT



RAW



RLE - Pro e Contro

Adatta per:

- immagini artificiali o con poco livello di dettaglio

Non adatta per:

- testi
- immagini con molte sfumature di colore o alto livello di dettaglio

Immagine ingrandita di un file di 16 x 16 pixel, costituito da 256 colori unici differenti.

Questo file, salvato in formato BMP non compresso, occupa 822 byte. Salvato invece sempre in formato BMP, ma utilizzando l'algoritmo RLE, occupa 1400 byte, cioè 1,7 volte la sua grandezza originale.

Entropia

Entropia di una sorgente di informazioni è una misura della *quantità di informazione* contenuta nel flusso di bit che deve essere sottoposto a compressione = numero di bit (teorico) necessari per codificarla

$$H(S) = \eta = \Sigma_i \, p_i \log_2 (1/p_i)$$

- p_i è la probabilità che il valore i-esimo si presenti
- $\log_2 1/p_i$ è il minimo numero di bit necessario per codificare in modo riconoscibile il valore i-esimo

Codifica Entropica

Si basa sulla probabilità che un certo valore appaia all'interno di una sequenza, e codifica i valori con un numero di bit diverso in funzione della frequenza

Esempio:

- in un'immagine con una distribuzione uniforme di toni di grigio p_i = 1/256, quindi servono 8 bit per codificare ogni tono di grigio

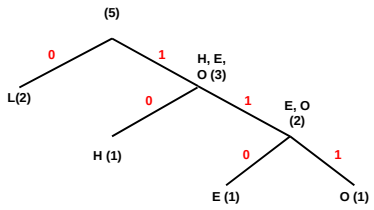
Dal teorema di Shannon si ricava che, per la compressione lossless, la lunghezza media di un codice (e quindi il rapporto di compressione ottenibile), è limitata inferiormente dall'entropia della sorgente:

$$H(S) \leq L$$

Algoritmo di Shannon-Fano

Crea un albero binario in modo top-down. I nodi dell'albero sono i simboli o gruppi di simboli:

- ordina i nodi in base al numero delle occorrenze
- dividi ricorsivamente l'insieme dei nodi in due parti, ognuna contenente, approssimativamente, lo stesso numero di occorrenze, finchè ogni parte contiene un solo simbolo



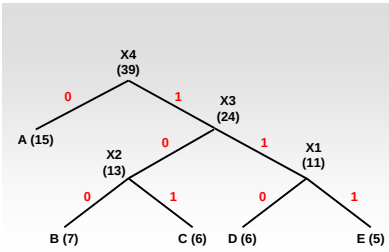
H	E	L	O
1	1	2	1
10	110	0	111

N. di Bit = 10 (teorico 9.6)

Codifica di Huffmann

Crea un albero binario in modo bottom-up. I nodi dell'albero sono simboli o gruppi di simboli

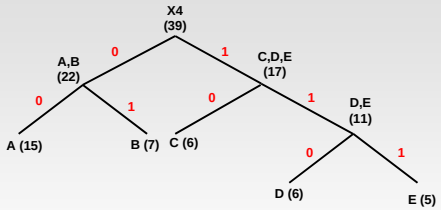
- ad ogni passo prende i due nodi con le probabilità più basse, crea un sottoalbero e gli assegna come probabilità la somma delle probabilità dei nodi figli
- etichetta i rami di destra con il bit 1, quelli di sinistra con il bit 0
- il codice di ogni simbolo è l'etichetta del cammino dalla radice



simbolo	#	$\log_2(1/p_i)$	codice	# di bit
A	15	1.38	0	15
B	7	2.48	100	21
C	6	2.70	101	18
D	6	2.70	110	18
E	5	2.96	111	15

N. di bit = 87 (teorico 85.29)

Huffman vs. Shannon-Fano



simbolo	#	$\log_2(1/p_i)$	codice	# di bit
A	15	1.38	00	30
B	7	2.48	01	14
C	6	2.70	10	12
D	6	2.70	110	18
E	5	2.96	111	15

N. di bit = 89 (87, Huffman, teorico 85.29)

Codifica LZW

Costruisce dinamicamente il vocabolario dei simboli codificati

Codici fissi per sequenze di lunghezza variabile

```
w = NULL;
while (not EOF)
 { read a character k
   if wk exists in the dictionary
      w = wk;
   else
    { add wk to the dictionary;
      output the code for w;
      w = k;
    }
 } output the code for w;
```

w	k	output	codice	simbolo
NULL	a			

Decodifica LZW

La decodifica ricostruisce il vocabolario mentre espande il testo

```
read a character k;
output k;
w = k;
while ( read a symbol k )
 { entry = dictionary entry for k;
   output entry;
   add w + entry[0] to dictionary;
   w = entry;
 }
```

w	k	output	codice	simbolo
	a	a		

Caso particolare

Prendiamo la stringa:
ababbabcabbabbax

```
w = NULL;
while (not EOF)
 { read a character k
   if wk exists in the dictionary
      w = wk;
   else
    { add wk to the dictionary;
      output the code for w;
      w = k;
    }
 } output the code for w;
```

w	k	output	codice	simbolo
NULL	a			
a	b	a	256	ab
b	a	b	257	ba
a	b			
ab	b	256	258	abb
b	a			
ba	b	257	259	bab
b	c	b	260	bc
c	a	c	261	ca
a	b			
ab	b			
abb	a	258	262	abba
a	b			
ab	b			
abb	a			
abba	x	262	263	abbax
x	EOF	X		

Fallimento della decodifica

Prendiamo la stringa: ab 256 257 bc 258 262 x

```
read a character k;
output k;
w = k;
while ( read a symbol k )
  { entry = dictionary entry for k;
    output entry;
    add w + entry[0] to dictionary;
    w = entry;
  }
```

w	k	output	codice	simbolo
	a	a		

Soluzione

La concatenazione di *carattere + stringa + carattere* fa fallire il decoder (a + bb + a + …)

```
read a character k;
output k;
w = k;
while ( read a symbol k )
  { entry = dictionary entry for k;
    /*Gestione Eccezione */
    if (entry == null)
          entry = w + w[0];
    output entry;
    add w + entry[0] to dictionary;
    w = entry;
  }
```

w	k	output	codice	simbolo
	a	a		
a	b	b	256	ab
b	256	ab	257	ba
ab	257	ba	258	abb
ba	b	b	259	bab
b	c	c	260	bc
c	258	abb	261	ca
abb	262	abba	262	abba
abba	x	x	263	abbax
x	EOF