Università degli Studi di Bergamo

DIPARTIMENTO DI INGEGNERIA E SCIENZE APPLICATE

RETI INTERNET MULTIMEDIALI

Codifica Numerica e Trasformate Discrete

Il documento è adattato da materiale cortesemente messo a disposizione dal Prof. Stefano Paris e dal Prof. Vittorio Trecordi

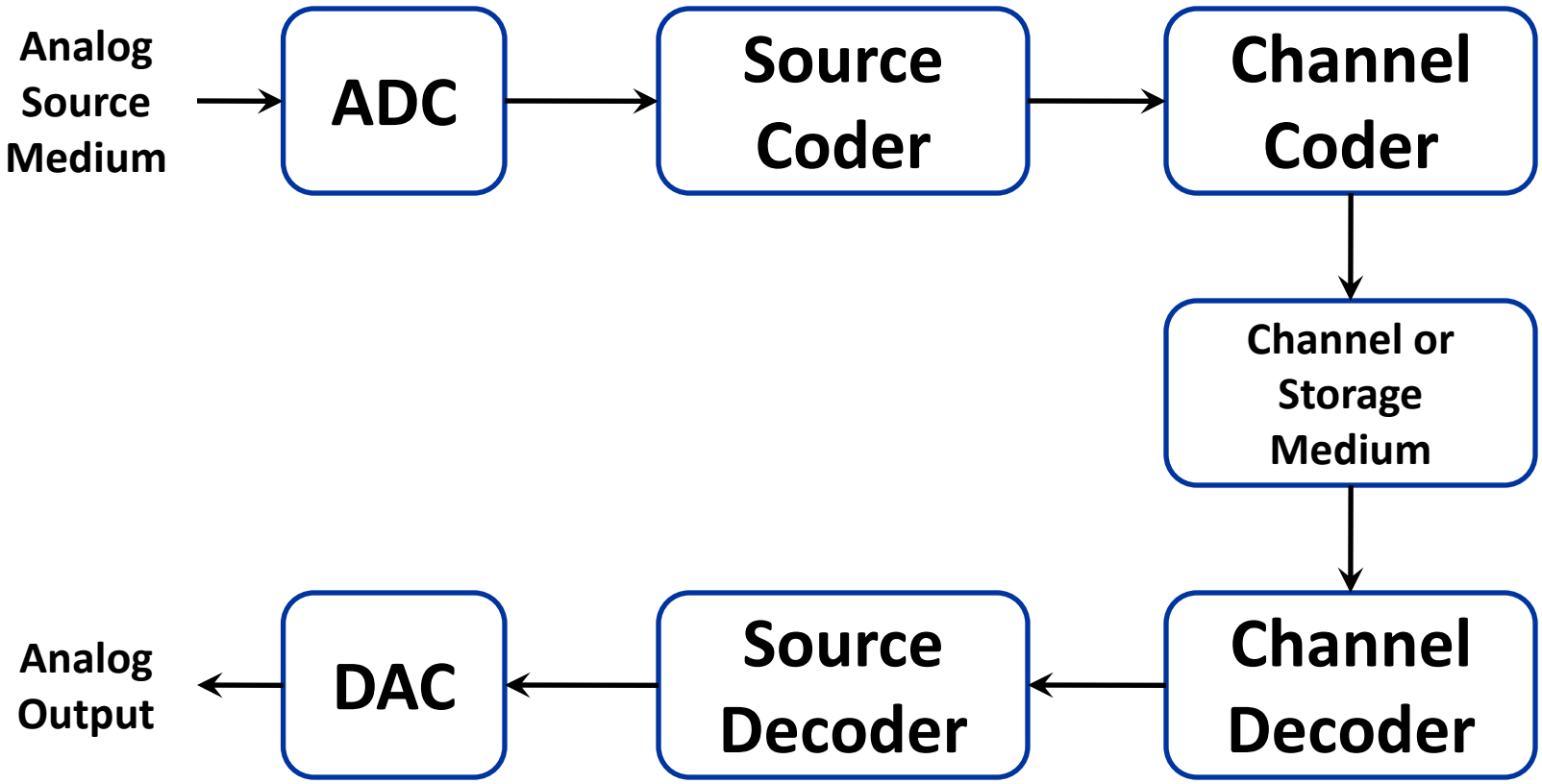
CODIFICA NUMERICA

Introduzione

Introduzione alla Codifica Numerica

- Applicazioni e tecnologie multimediali si basano su audio, immagini e video *digitali* (o *numerici)*
- Una rappresentazione numerica dell'informazione consente
  - Una maggiore flessibilità
  - L'introduzione di nuovi servizi
- La codifica numerica consente di sfruttare appieno i vantaggi resi possibili dall'introduzione della rappresentazione digitale dell'informazione
  - Permette di rappresentare l'informazione digitale in modo efficiente, andando ad eliminare ridondanza informativa

Modello di un sistema di comunicazione



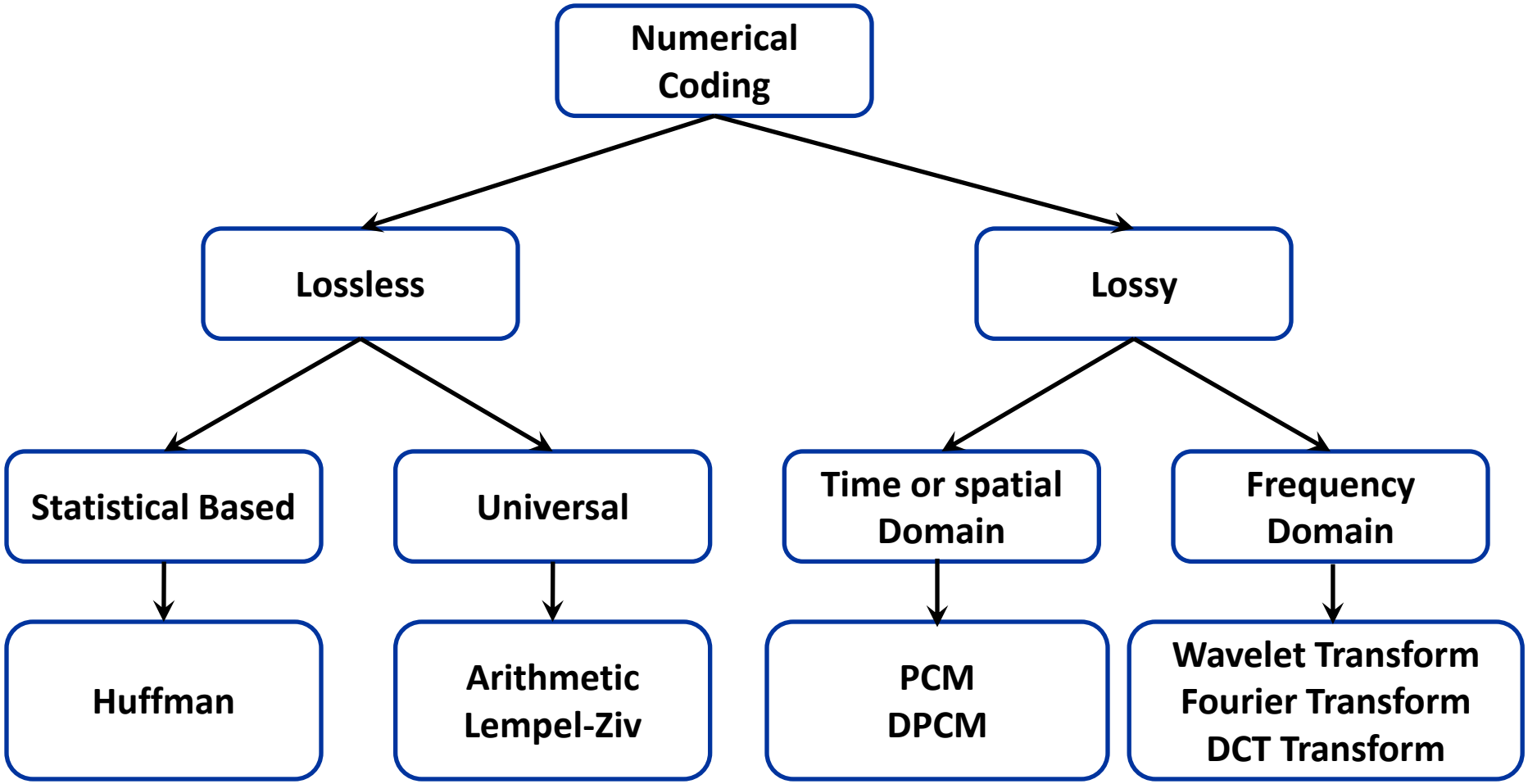
Codifica di sorgente e codifica di canale

- La codifica viene eseguita in diversi punti della trasmissione di contenuti multimediali
  - Codifica di sorgente (source coding): il contenuto originale viene compresso in un file (o stream)
  - Codifica di canale (channel coding): i dati (file o stream) sono convertiti in un segnale adatto alla trasmissione sul canale di comunicazione
- I due tipi di codifica hanno obiettivi che determinano effetti opposti dal punto di vista della dimensione dei dati
  - Source coding: comprimere i dati (riduce la ridondanza dell'informazione) per minimizzare la quantità di dati da trasferire
  - Channel coding: garantire affidabilità nella trasmissione dei dati (aumenta la ridondanza dell'informazione per proteggere la trasmissione da eventuali errori)

Codifica lossless e Codifica lossy

- La codifica numerica (di sorgente) viene utilizzata in numerosi settori dell'ICT
- Esistono due diversi tipi di codifica numerica
  - Codifica lossless: non si ha perdita del contenuto informativo originario
  - Codifica lossy: si ha perdita parziale del contenuto informativo originario
- In caso di codifica lossy, parte del contenuto informativo viene intenzionalmente perso per ridurre la dimensione del contenuto codificato
  - Esiste un trade-off tra la dimensione del contenuto codificato e la sua qualità

Codifica lossless e Codifica lossy

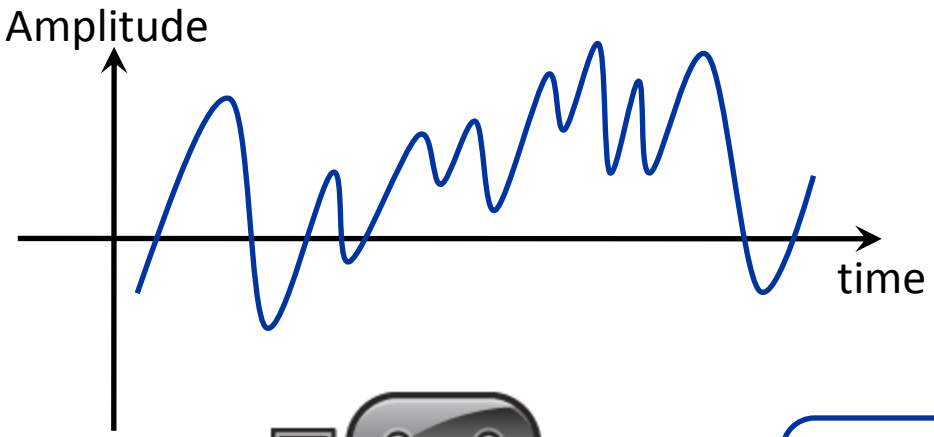


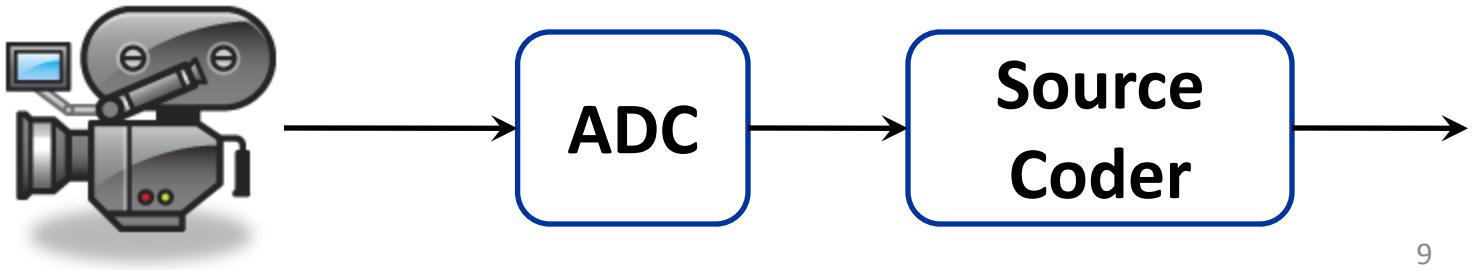
Caratteristiche della Codifica Numerica

- La codifica utilizzata e i relativi parametri per una specifica applicazione sono scelti considerando
  - Efficienza (rapporto di compressione)
  - Ritardo
  - Complessità
  - Qualità dell'informazione decodificata
- Diversi tipi di codifica possono produrre come output diverse tipologie di bitrate
  - Bitrate costante (CBR)
  - Bitrate variabile (VBR)

Caratteristiche della Codifica Numerica

- Codifica temporale
  - Il segnale in ingresso varia nel tempo
  - Es: voce, video

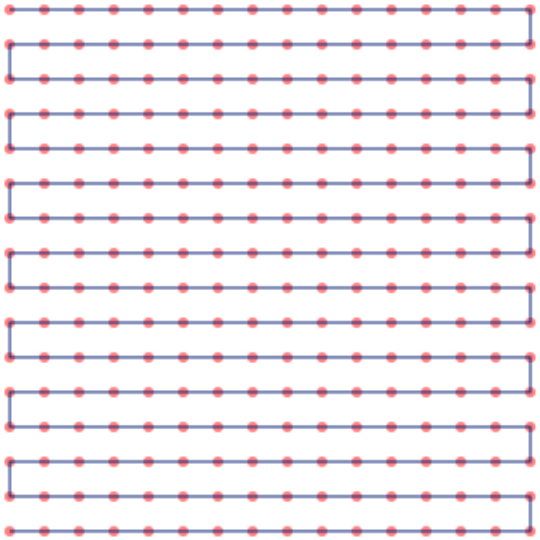




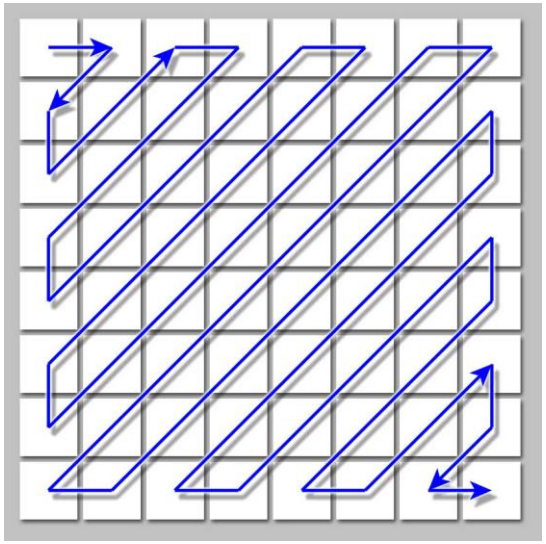
Caratteristiche della Codifica Numerica

- Codifica spaziale
  - Il segnale varia in una o più dimensioni dello spazio
  - Es: immagine, video
  - La sequenza spaziale (il percorso) utilizzata dalla codifica viene definita per convenienza a priori

Linear



Zig-Zag



Concetti fondamentali

- Per misurare le prestazioni degli algoritmi di codifica utilizzati nella compressioni di contenuti multimediali è necessario introdurre alcuni concetti fondamentali della teoria dell'informazione
  - Informazione
  - Entropia
- Tali concetti permettono di calcolare la quantità di informazione di una sorgente

Informazione: assunzioni

- L’informazione di un messaggio/evento è tanto maggiore quanto meno probabile è il messaggio/evento
- L’informazione di una coppia di messaggi indipendenti è la somma delle rispettive quantità di informazione
- Nota: solo il contenuto sintattico è di interesse ai fini della misura di informazione (non quello semantico)

Informazione

- Sia x_i un messaggio generato dalla sorgente X con probabilità $P(x_i)$
- L'informazione del messaggio x_i è definita come

$$I(x_i) = \log \frac{1}{P(x_i)} = -\log P(x_i)$$

- La base del logaritmo identifica l'unità di misura
  - Per messaggi rappresentati in forma binaria il logaritmo è in base 2 e $I(x_i)$ si misura in bit

$$I(x_i) = \log_2 \frac{1}{P(x_i)} = -\log_2 P(x_i) \text{ [bit]}$$

Entropia

- L'Entropia della sorgente X è definita come il valore atteso dell'informazione della sorgente X

$$H(X) = \sum_{i=1}^{N} P(x_i) I(x_i) = -\sum_{i=1}^{N} P(x_i) \log P(x_i)$$

- L'entropia rappresenta perciò una misura
  - Del contenuto informativo atteso della sorgente
  - Dell'incertezza media della sorgente
- Se la base del logaritmo è 2 l'entropia viene misurata in bit (logaritmo naturale → nat)

Entropia

- Si consideri una variabile casuale X in {0,1}

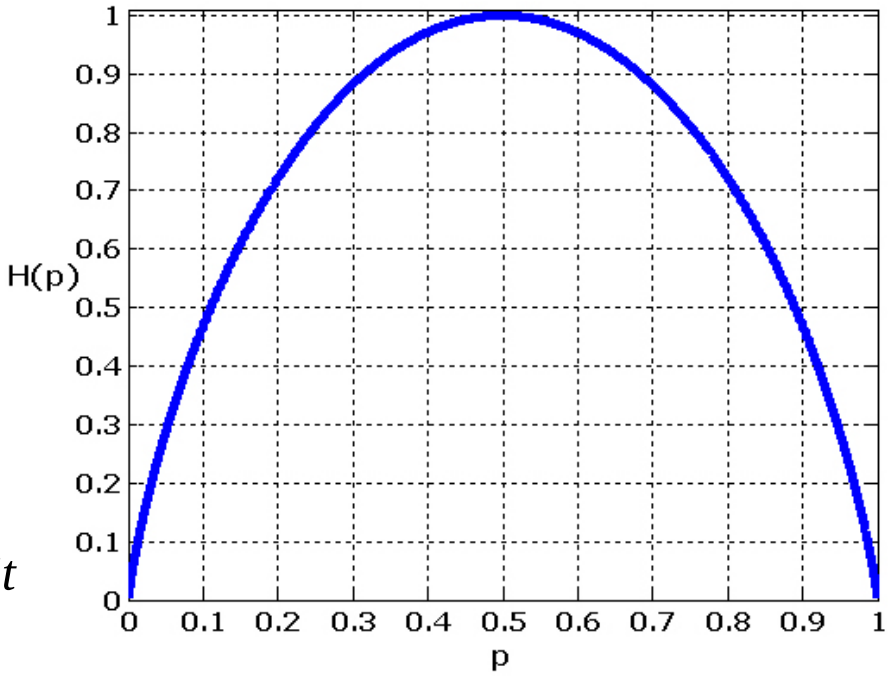
$$X = \begin{cases} 1 & \text{con prob. } p \\ 0 & \text{con prob. } 1-p \end{cases}$$

$p = 0.5$

$H(X) = -(0.5\log 0.5 + 0.5\log 0.5) = 1 \quad bit$

$p = 0.9$

$H(X) = -(0.1\log 0.1 + 0.9\log 0.9) = 0.469 \quad bit$

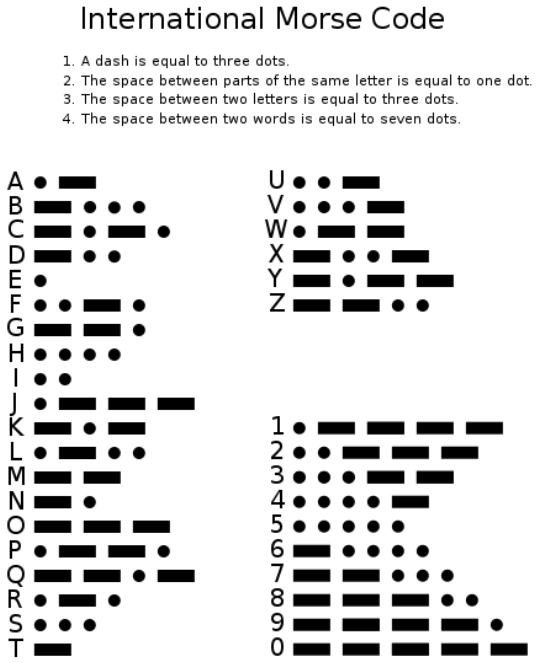


CODIFICA NUMERICA

Codifica Lossless

Lossless Entropy Coding

- Tecniche di codifica che utilizzano modelli statistici basati sull'occorrenza dei simboli di input
  - Esempio Classico: Codice Morse
- Dato un insieme di simboli in ingresso sono generate parole di codice (o simboli di output) di lunghezza variabile tali da minimizzare la lunghezza attesa delle stesse parole di codice
- L'obiettivo è quindi quello di ridurre ridondanza e ottenere un valor medio di informazione simile all'entropia della sorgente
  - Per fare ciò, i simboli a maggiore occorrenza vengono codificati con parole di codice di lunghezza inferiore



Letter	probability German	probability English	Morse Code
e	16,65%	12,41%	·
n	10,36%	6,41%	–·
i	8,14%	6,46%	··
t	5,43%	8,90%	–
a	5,15%	8,09%	·–
o	2,25%	8,13%	–––
x	0,03%	0,20%	–··–
y	0,03%	2,14%	–·––

Codifica di Huffman

- Utilizzato negli algoritmi di compressione di tipo Lossless
- Assume che la distribuzione di probabilità dei simboli alla sorgente sia conosciuta
- L'idea è quella di creare un albero binario in cui la somma della lunghezza dei percorsi per raggiungere le foglie è minima
- L'algoritmo costruisce l'albero a partire dalle foglie (bottom-up), che rappresentano i simboli generati dalla sorgente

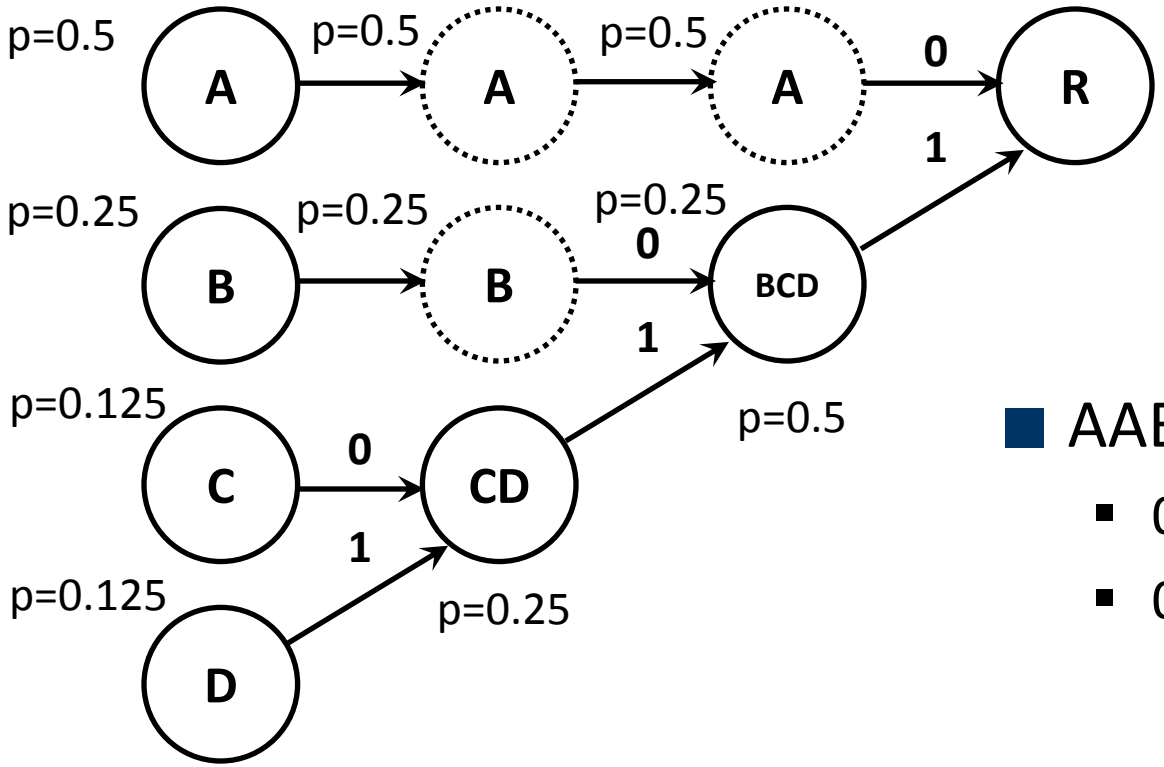
Codifica di Huffman

- Algoritmo:
  1. Ordina i simboli in ordine decrescente rispetto alle loro probabilità
  2. Combina gli ultimi due simboli in un nuovo simbolo. Ripeti il passo (1) finché la nuova lista contiene solo 2 simboli (figli della radice)
  3. A partire dal nodo radice assegna al ramo sx ‘0’ e al dx ‘1’
- Il codice di ogni simbolo è uguale alla sequenza di ‘1’ e ‘0’ del percorso che unisce la radice con la foglia che rappresenta il simbolo

Codifica di Huffman

- Esempio:

A	B	C	D
0.5	0.25	0.125	0.125

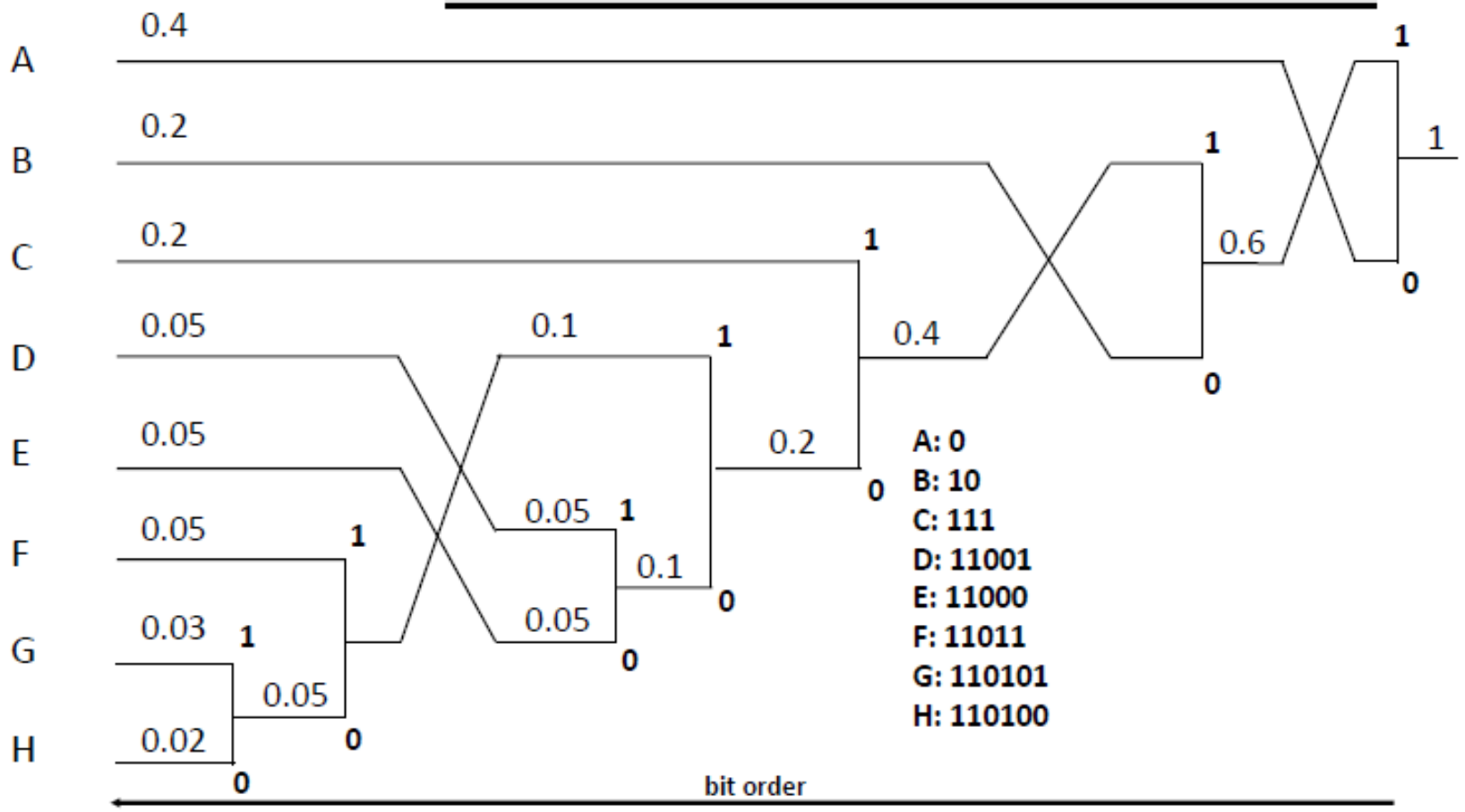


	Huffman	Binary
A	0	00
B	10	01
C	110	10
D	111	11

- AABAABCD è codificato:
  - 00100010110111 (14 bits)
  - 0000010000011011 (16 bits)

Codifica di Huffman - Esempio

A	B	C	D	E	F	G	H
0.4	0.2	0.2	0.05	0.05	0.05	0.03	0.02



A: 0
B: 10
C: 111
D: 11001
E: 11000
F: 11011
G: 110101
H: 110100

Codifica di Huffman

- Esistono casi in cui la codifica di Huffman non determina univocamente la lunghezza delle parole di codice, a causa delle scelte arbitrarie fra coppie di probabilità minime
  - Per esempio nel caso di una sorgente con probabilità {0.4, 0.2, 0.2, 0.1, 0,1} è possibile ottenere parole di codice di lunghezza [2, 2, 2, 3, 3] e [1, 2, 3, 4, 4]
- Sarebbe meglio avere parole di codice la cui lunghezza ha varianza minima
  - Soluzione più efficiente in quanto necessita di buffer di lunghezza minima per il trasmittente e per il ricevente
- Il codice di Huffman è «prefix-free»
  - Nessuna parola di codice è il prefisso di altre parole

Efficienza del codice

- Codifica in assenza di rumore (*1° Teorema di Shannon*)
  - Per una sorgente di informazione discreta e senza memoria, il valore minimo teorico della lunghezza di codice è pari all'entropia della sorgente
- E' quindi possibile stabilire una misura dell'*efficienza* di una tecnica di codifica a lunghezza variabile

$$\eta = \frac{H}{L_{avg}}$$

- H è l'entropia della sorgente, mentre L_{avg} è la lunghezza media delle parole usate dal codice
- La codifica entropica cerca di ottenere valori di efficienza η prossimi a 1

Codifica di Huffman

- La Codifica di Huffman è ottima quando la probabilità di ciascun simbolo in ingresso è una potenza negativa di 2
  - In questo caso l'efficienza del codice è pari a 1
- Se un simbolo dell'alfabeto ha una probabilità molto prossima ad 1 e gli altri simboli una probabilità molto bassa, la lunghezza media del codice è 1 bit/simbolo poichè la lunghezza minima per una parola di codice è 1 bit, mentre l'entropia è di molto inferiore
- Una variante è la **Codifica Adattativa di Huffman**
  - Non è necessaria una conoscenza pregressa della distribuzione statistica dei simboli alla sorgente
  - Il modello statistico viene continuamente aggiornato mentre l'algoritmo procede alla codifica dei simboli
  - Più flessibile, poichè si adatta ad un eventuale cambiamento nella statistica dei simboli

Codifica Aritmetica

- La Codifica di Huffman porta a risultati subottimi perché esiste una corrispondenza biunivoca tra simboli e parole di codice
- La Codifica Aritmetica usa una sola parola di codice per una sequenza di simboli lunga m
  - La limitazione relativa all'assegnamento di un numero intero di bit per simbolo propria della Codifica di Huffman viene risolta

Codifica Aritmetica

- Algoritmo di codifica
  1. Inizializza l'**intervallo corrente** a [0,1)
  2. Per ogni carattere della sequenza
     a) **Suddividi** l'intervallo corrente **in sottointervalli**, uno per ciascun simbolo dell'alfabeto. **L'ampiezza** di ciascun sottointervallo è **proporzionale alla probabilità** stimata che il simbolo corrente sia uguale al simbolo corrispondente all'intervallo
     b) **Seleziona** come intervallo corrente il **sottointervallo** corrispondente al **carattere corrente**
  3. La sequenza è codificata con il **lower bound** dell'intervallo risultante al termine delle operazioni effettuate al punto 2
     - La lunghezza finale di tale intervallo è uguale al prodotto delle probabilità dei singoli caratteri della sequenza codificata

Codifica Aritmetica

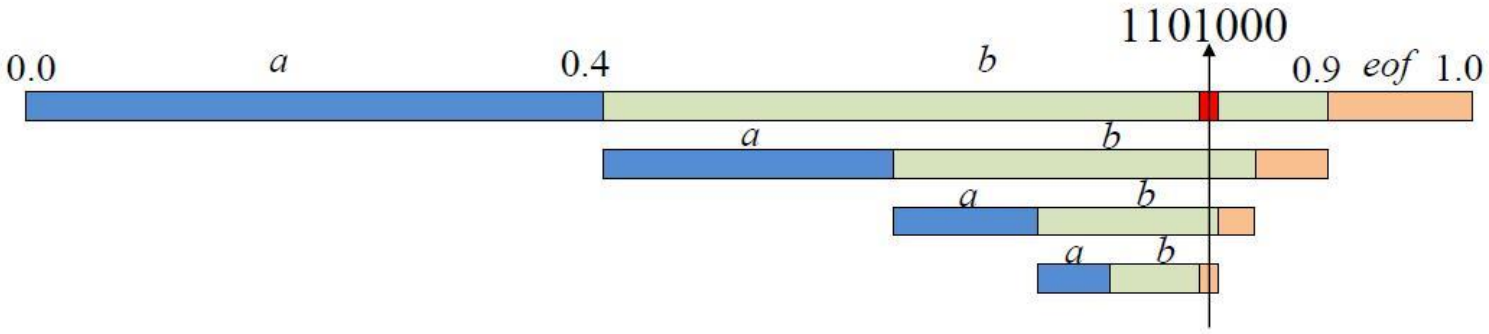
- Esempio (codifica)

A	B	C
0.4	0.5	0.1

- Input: BBBC

Intervallo Corrente	A	B	C	Input	Azione
[0,1)	[0, 0.4)	**[0.4, 0.9)**	[0.9, 1)	B	Dividi
[0.4, 0.9)	[0.4, 0.6)	**[0.6, 0.85)**	[0.85, 0.9)	B	Dividi
[0.6, 0.85)	[0.6, 0.7)	**[0.7, 0.825)**	[0.825, 0.85)	B	Dividi
[0.7, 0.825)	[0.7, 0.75)	[0.75, 0.8125)	**[0.8125, 0.825)**	C	Stop
[0.8125, 0.825)					

Codifica Aritmetica



- L'intervallo finale è [0.8125,0.825), che è approssimato in rappresentazione binaria [0.1101, 0.11010011)
- L'intervallo è identificato dal suo lower bound
  - Si può quindi trasmettere la sola stringa **1101**

Codifica Aritmetica

- Il decodificatore ha bisogno delle probabilità dei simboli codificati, in quanto esegue simili operazioni a quelle della codifica
- Algoritmo di decodifica
  1. Inizializza l'intervallo corrente a [0,1)
  2. Per ciascuno degli N caratteri da decodificare
     a. Suddividi l'intervallo corrente in sottointervalli, uno per ciascun simbolo dell'alfabeto. L'ampiezza di ciascun sottointervallo è proporzionale alla probabilità stimata che il carattere corrente sia uguale al simbolo corrispondente all'intervallo
     b. Seleziona il carattere corrispondente al sottointervallo in cui cade il valore x e seleziona tale sottointervallo come intervallo corrente
        - x è la parola ottenuta dal carattere attuale e da tutti i caratteri precedenti (si vanno a considerare cifre decimali sempre meno significative)

Codifica Aritmetica

- Esempio (decodifica)

A	B	C
0.4	0.5	0.1

- Input: 0.1101 ➔ 0.8125

Intervallo Corrente	A	B	C	Input	Output
[0,1)	[0, 0.4)	**[0.4, 0.9)**	[0.9, 1)	0.8	B
[0.4, 0.9)	[0.4, 0.6)	**[0.6, 0.85)**	[0.85, 0.9)	0.81	B
[0.6, 0.85)	[0.6, 0.7)	**[0.7, 0.825)**	[0.825, 0.85)	0.812	B
[0.7, 0.825)	[0.7, 0.75)	[0.75, 0.8125)	**[0.8125, 0.825)**	0.8125	C

Codifica Lempel Ziv

- Algoritmo di codifica indipendente dalla statistica della sorgente
  - Codificatori di sorgente universali
- Algoritmo di compressione adattativo basato sul concetto di *dizionario*
  - Si costruisce automaticamente un dizionario che comprenda le stringhe viste fino ad adesso nel messaggio da codificare
  - Il testo precedente costituisce infatti un ottimo dizionario dato che generalmente condivide lo stesso linguaggio e stile della restante parte del messaggio da codificare

Codifica Lempel Ziv (LZ77)

- Algoritmo di codifica
  1. La sequenza d'ingresso è iterativamente scomposta in blocchi di lunghezza variabile, chiamati **frasi**
  2. Ogni volta che un blocco di lettere differisce da uno dei blocchi precedenti per l'ultima lettera esso viene inserito nel dizionario
  3. Tutte le frasi sono riportate nel dizionario unitamente all'informazione sulle locazioni in cui ciascuna frase compare
  4. Nel codificare quindi una nuova frase, l'algoritmo registra nel dizionario la locazione da cui essa inizia e la nuova lettera

Codifica Lempel Ziv (LZ77)

- Il messaggio inviato consiste in una serie di triplette <P, L, C>
  - P (*puntatore*) indica quanto indietro guardare nel testo già decodificato
  - L = la lunghezza della frase
  - C = carattere successivo nel messaggio di input

Codifica Lempel Ziv (LZ77)

- Il dizionario non deve essere inviato
- Il decodificatore lo costruisce utilizzando lo stesso meccanismo utilizzato dal codificatore
- Esistono numerosi varianti dell'algoritmo
  - Rappresentazione dei puntatori
  - Lunghezza massima oltre la quale i puntatori non possono più fare riferimento (*finestra*)
  - Ampiamente usata: GIF, TIFF, PDF…
- Due tra le più importanti sono chiamate LZ77 e LZ78

Codifica Lempel Ziv (LZ77)

- Esempio (codifica)

  Alfabeto = {x, y}

  Input: x y xx yxy xxyy

  Output: <0,0,x> <0,0,y> <2,1,x> <3,2,y> <5,3,y>

- Esempio (decodifica)

  Alfabeto = {x, y, z}

  Input: <0,0,x> <0,0,y> <2,1,z> <2,1,x> <5,3,z> <6,3,z> <5,2,z>

  Output: x y xz xx yxzz xxyz zxz

Codifica Lempel Ziv (LZ78)

- In caso di ricorrenza di una frase, non è necessario comunicare la lunghezza della frase ricorrente
  - Viene rimosso il concetto di *finestra*
- La parola di codice è ottenuta concatenando
  - La locazione del dizionario che contiene la frase precedente che coincide fino al penultimo carattere
    - 0000 è il codice utilizzato per indicare una locazione nulla
  - Il nuovo carattere
- NB: Il puntatore in LZ78 punta a una entry del dizionario, mentre in LZ77 punta a un simbolo del testo già codificato!
- Esempio
  - Input: 1010110100100111010100001100111010110001101 1
  - Scomposizione in frasi: 1, 0, 10, 11, 01, 00, 100, 111, 010, 1000, 011, 001, 110, 101, 10001, 1011

	Locazione dizionario	Contenuto dizionario	Parola di codice
1	0001	1	0000,**1**
2	0010	0	0000,**0**
3	0011	10	0001,**0**
4	0100	11	0001,**1**
5	0101	01	0010,**1**
6	0110	00	0010,**0**
7	0111	100	0011,**0**
8	1000	111	0100,**1**
9	1001	010	0101,**0**
10	1010	1000	0111,**0**
11	1011	011	0101,**1**
12	1100	001	0110,**1**
13	1101	110	0100,**0**
14	1110	101	0011,**1**
15	1111	10001	1010,**1**
16		1011	1110,**1**

CODIFICA NUMERICA

Trasformate Discrete

Scomposizione di un segnale

- Lo studio dell'argomento risale all'inizio del secolo XIX (1807, J. Fourier)
  - Un segnale periodico può essere espresso come combinazione lineare di un numero finito di sinusoidi (di frequenza, ampiezza e fase differente)
  - Un segnale non periodico può essere espresso come combinazione lineare di un numero infinito di sinusoidi (di frequenza, ampiezza e fase differente)

Trasformate Discrete

- Le trasformate discrete sono utilizzate per rappresentare nel dominio delle frequenze un segnale campionato nel dominio del tempo o dello spazio
- Il segnale è rappresentato come somma pesata di funzioni base
- La sequenza dei *coefficienti* in frequenza permette di ricreare il segnale originario nel dominio del tempo o dello spazio
  - Coefficienti “non utili” ai fini dell’utilizzo del segnale originario possono essere filtrati (es. coefficienti relativi a componenti non percettibili dall’uomo)

Discrete Fourier Transform (DFT)

- Rappresentazione di una sequenza di campioni di un segnale unidimensionale nel dominio delle frequenze
- E' utilizzata per l'analisi di segnali a tempo discreto composti da un numero finito di campioni
- Il segnale è la somma di N funzioni sinusoidali (funzioni base)

Discrete Fourier Transform (DFT)

- DFT in una dimensione di un segnale $x(k)$ composto da N campioni:

$$X(n) = \frac{1}{N}\sum_{k=0}^{N-1} x(k) e^{\left(\frac{-j2\pi nk}{N}\right)} \quad n = 0,1,2\ldots, N-1$$

$$= \frac{1}{N}\sum_{k=0}^{N-1} x(k)\left[\cos\left(2\pi nk\right) - j\sin\left(2\pi nk\right)\right]$$

- Inverse DFT (IDFT) 1-D:

$$x(k) = \sum_{n=0}^{N-1} X(n) e^{\left(\frac{j2\pi nk}{N}\right)} \quad k = 0,1,2\ldots, N-1$$

Discrete Fourier Transform (DFT)

- DFT in due dimensioni con NxM campioni:

$$X(n,m) = \frac{1}{NM} \sum_{k=0}^{N-1} \sum_{h=0}^{M-1} x(k,h) e^{\left(\frac{-j2\pi nk}{N}\right)} e^{\left(\frac{-j2\pi mh}{M}\right)} \qquad \begin{matrix} n = 0,1,2\ldots, N-1 \\ m = 0,1,2\ldots, M-1 \end{matrix}$$

- DFT 2-D inversa:

$$x(k,h) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} X(n,m) e^{\left(\frac{j2\pi nk}{N}\right)} e^{\left(\frac{j2\pi mh}{M}\right)} \qquad \begin{matrix} k = 0,1,2\ldots, N-1 \\ h = 0,1,2\ldots, M-1 \end{matrix}$$

DFT - Proprietà

- **Linearità**:

  Se $x(k)$ e $y(k)$ sono due successioni di valori reali di lunghezza N, le cui rispettive DFT sono $X(n)$ e $Y(n)$, allora

$$x(k) \xrightarrow{DFT} X(n) \quad y(k) \xrightarrow{DFT} Y(n)$$

$$ax(k)+by(k) \xrightarrow{DFT} aX(n)+bY(n)$$

- **Periodicità:**

  $x(k)$ e $X(n)$ sono periodiche di periodo N

DFT - Proprietà

- **Simmetria**:

  Se $x(k)$ è una successione di valori reali di lunghezza N, la cui DFT è $X(n)$, allora

$$X(n) = \overline{X}(-n)$$

Dalla relazione precedente segue

$\Re(\{X(n)\})$ Funzione pari

$\Im(\{X(n)\})$ Funzione dispari

DFT - Proprietà

- **Traslazione**:

  Se $x(k)$ è una successione di valori reali di lunghezza N, la cui DFT è $X(n)$, allora

$$x(k) \xrightarrow{DFT} X(n)$$
$$x(k-k_0) \xrightarrow{DFT} e^{-jnk_0} X(n)$$
$$e^{-jn_0k} x(k) \xrightarrow{DFT} X(n-n_0)$$

  con k_0 e n_0 costanti

Discrete Fourier Transform (DFT)

- Il calcolo della DFT è molto dispendioso dal punto di vista computazionale: $O(N^2)$, dovuto al numero di moltiplicazioni da effetturare per il calcolo della trasformata
- Può essere eseguito in modo efficiente utilizzando una delle molte versioni dell'algoritmo FFT (Fast Fourier Transform)
  - Decimazione nel tempo: due sommatorie
    - Somma delle componenti $x(k)$ con k pari
    - Somma delle componenti $x(k)$ con k dispari
  - Decimazione nelle frequenze
    - Calcolo delle componenti in frequenza $X(n)$ con n pari
    - Calcolo delle componenti in frequenza $X(n)$ con n dispari

Fast Fourier Transform (FFT)

- Consideriamo l'algoritmo **radix-2 FFT con decimazione nel tempo**
- L'obiettivo è ridurre il numero di moltiplicazioni per il calcolo della DFT
- Definiamo $W_N = e^{-j\frac{2\pi}{N}}$
- Riscriviamo la DCT:
  $X(n) = \sum_{k=0}^{N-1} x(k) W_N^{kn} \quad (n = 0,1, \dots, N-1)$

Fast Fourier Transform (FFT)

- Consideriamo le seguenti proprietà:
  - Periodicità: $W_N^{k+N} = W_N^k$
  - Simmetria: $W_N^{k+\frac{N}{2}} = -W_N^k$
- Scomponiamo $X(n)$ in due sommatorie, una riguardante i campioni pari e una quelli dispari:

$$X(n) = \sum_{k=0}^{\frac{N}{2}-1} \underbrace{x(2k)W_N^{2kn}}_{\text{Pari}} + \sum_{k=0}^{\frac{N}{2}-1} \underbrace{x(2k+1)W_N^{(2k+1)n}}_{\text{Dispari}}$$

Fast Fourier Transform (FFT)

- Sviluppando i calcoli si ottiene:

$$X(n) = \sum_{k=0}^{\frac{N}{2}-1} x(2k) \cdot (W_N^2)^{kn} + W_N^n \sum_{k=0}^{\frac{N}{2}-1} x(2k+1) \cdot (W_N^2)^{kn}$$

- Considerando la seguente proprietà:

$$W_N^2 = e^{-j\left(\frac{2\pi}{N}2\right)} = e^{-j\left(\frac{\frac{2\pi}{N}}{2}\right)} = W_{\frac{N}{2}}$$

- Si ottiene l'espressione della FFT:

$$X(n) = \underbrace{\sum_{k=0}^{\frac{N}{2}-1} x(2k) \cdot (W_{\frac{N}{2}})^{kn}}_{\text{DFT degli } N/2 \text{ campioni pari } E(n)} + W_N^n \underbrace{\sum_{k=0}^{\frac{N}{2}-1} x(2k+1) \cdot (W_{\frac{N}{2}})^{kn}}_{\text{DFT degli } N/2 \text{ campioni dispari } O(n)}$$

Butterfly FFT

$$E(n) = \sum_{k=0}^{\frac{N}{2}-1} x(2k)\,(W_{\frac{N}{2}})^{kn} = E(n+\frac{N}{2})$$

$$O(n) = \sum_{k=0}^{\frac{N}{2}-1} x(2k+1)\,(W_{\frac{N}{2}})^{kn} = O(n+\frac{N}{2})$$

Proprietà di periodicità

$$X(n) = E(n) + W_N^n O(n)$$

$$X\left(n+\frac{N}{2}\right) = E(n) + W_N^{n+\frac{N}{2}} O(n)$$

$$= E(n) - W_N^n O(n)$$

Proprietà di simmetria

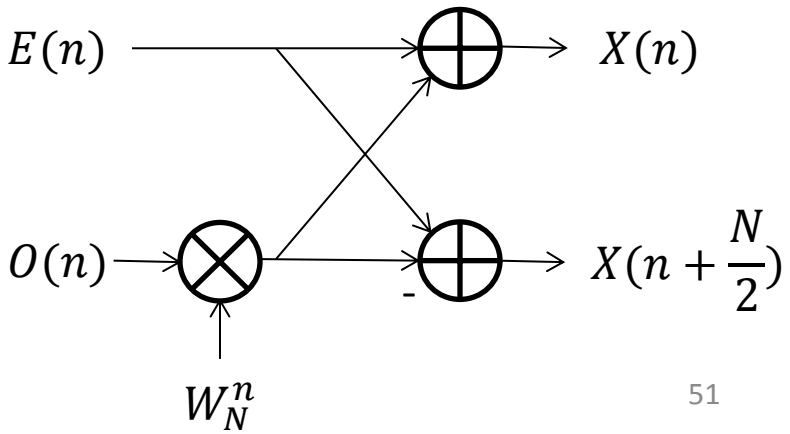
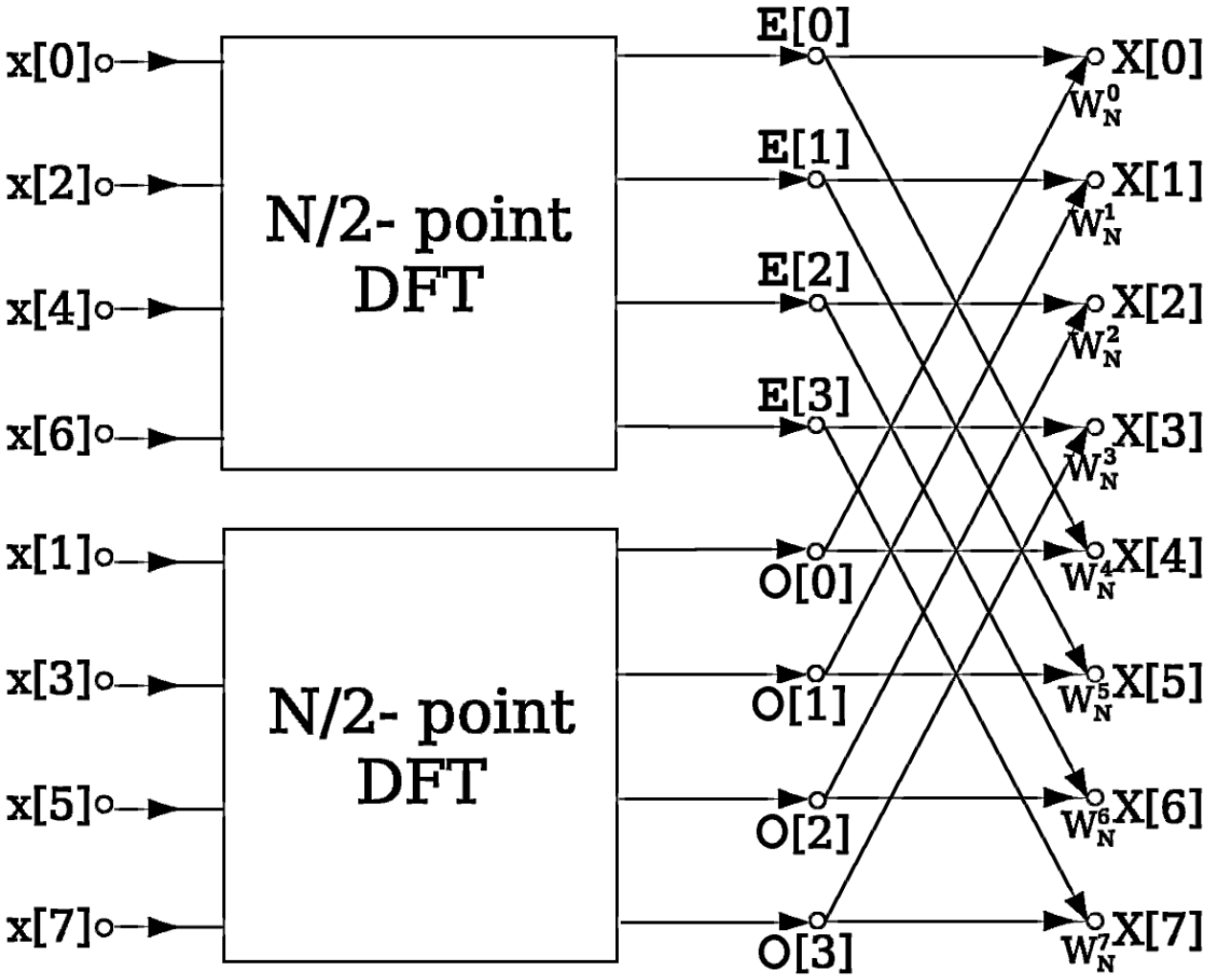


Diagramma a blocchi della FFT



Efficienza della FFT

$$X(n) = \underbrace{\sum_{k=0}^{\frac{N}{2}-1} x(2k) \cdot (W_{\frac{N}{2}})^{kn}}_{\left(\frac{N}{2}\right)^2 \text{ prodotti}} + \underbrace{W_N^n}_{\frac{N}{2} \text{ prodotti}} \underbrace{\sum_{k=0}^{\frac{N}{2}-1} x(2k-1) \cdot (W_{\frac{N}{2}})^{kn}}_{\left(\frac{N}{2}\right)^2 \text{ prodotti}}$$

- Per una DFT a 8 punti: $4^2 + 4 + 4^2 = 36$ prodotti invece di $8^2 = 64$ prodotti
  - Si risparmiano $64 - 36 = 28$ prodotti
- Per una DFT a 1000 punti: $500^2 + 500 + 500^2 = 500500$ prodotti invece di $1000^2 = 1000000$
  - Si risparmiano $1000000 - 500500 = 499500$ prodotti
- La complessità passa da $O(N^2)$ a $O(N \cdot \log_2 N)$ applicando ricorsivamente l'algoritmo

Discrete Cosine Transform (DCT)

- Trasformata molto utilizzata dagli algoritmi di compressione per immagini e video
- Utilizza come funzioni base cosinusoidi
- Rispetto alla DFT, la DCT è in grado di concentrare l'informazione utile in un numero inferiore di coefficienti
  - I coefficienti che trasportano poca informazione possono essere trascurati

Discrete Cosine Transform (DCT)

- DCT in una dimensione di un segnale:

$$X(n) = \alpha(n) \sum_{k=0}^{N-1} x(k) \cos\left(\frac{\pi(2k+1)n}{2N} \right) \quad n = 0,1,2,\ldots, N-1$$

- Inverse DCT 1-D:

$$x(k) = \sum_{n=0}^{N-1} \alpha(n) X(n) \cos\left(\frac{\pi(2k+1)n}{2N} \right) \quad k = 0,1,2,\ldots, N-1$$

- Multiplier:

$$\alpha(n) = \begin{cases} \sqrt{\frac{1}{N}} & n = 0 \\ \sqrt{\frac{2}{N}} & n \neq 0 \end{cases}$$

Discrete Cosine Transform (DCT)

- Il primo coefficiente ($n = 0$) è chiamato anche DC coefficient e rappresenta una “media” della sequenza di campioni

$$X(0) = \sqrt{\frac{1}{N}} \sum_{k=0}^{N-1} x(k)$$

- I restanti coefficienti sono detti coefficienti AC
- DC e AC per motivi storici (analisi di segnali in circuiti elettrici)
  - DC: Direct Current
  - AC: Alternate Current

Discrete Cosine Transform (DCT)

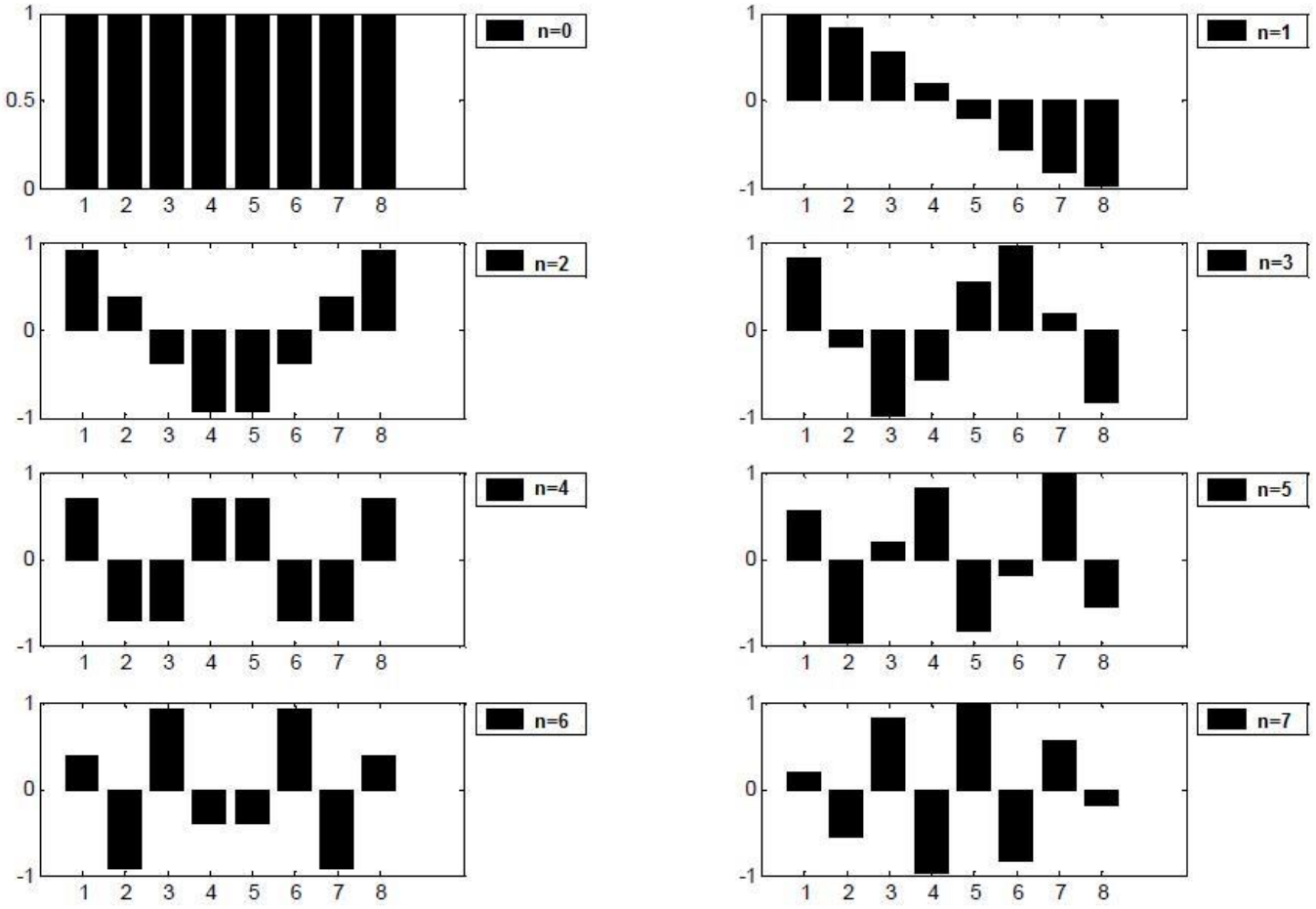
- Le waveforms (o basi) $WF(n)$ per $n = 0, 1, \dots, N-1$
  sono chiamate *Cosine Basis Function*

$$WF(n) = \cos\left(\frac{\pi(2k+1)n}{2N}\right) \quad k = 0,1, \dots, N-1$$

- Sono funzioni ortogonali
  - La moltiplicazione di due diverse WF, seguita dalla somma di tutti i campioni è nulla
  - La moltiplicazione di una WF con se stessa, seguita dalla somma di tutti i campioni è un valore non nullo

Discrete Cosine Transform (DCT)

- Basi ortogonali $WF(n)$ della 1-D DCT ($N = 8$)



Discrete Cosine Transform (DCT)

- DCT in due dimensioni con NxM campioni:

$$X(n,m) = \alpha(n)\alpha(m)\sum_{k=0}^{N-1}\sum_{h=0}^{M-1} x(k,h)\cos\left(\frac{\pi(2k+1)n}{2N}\right)\cos\left(\frac{\pi(2h+1)m}{2M}\right) \quad \begin{matrix} n = 0,1,2\ldots,N-1 \\ m = 0,1,2\ldots,M-1 \end{matrix}$$

- DCT 2-D inversa:

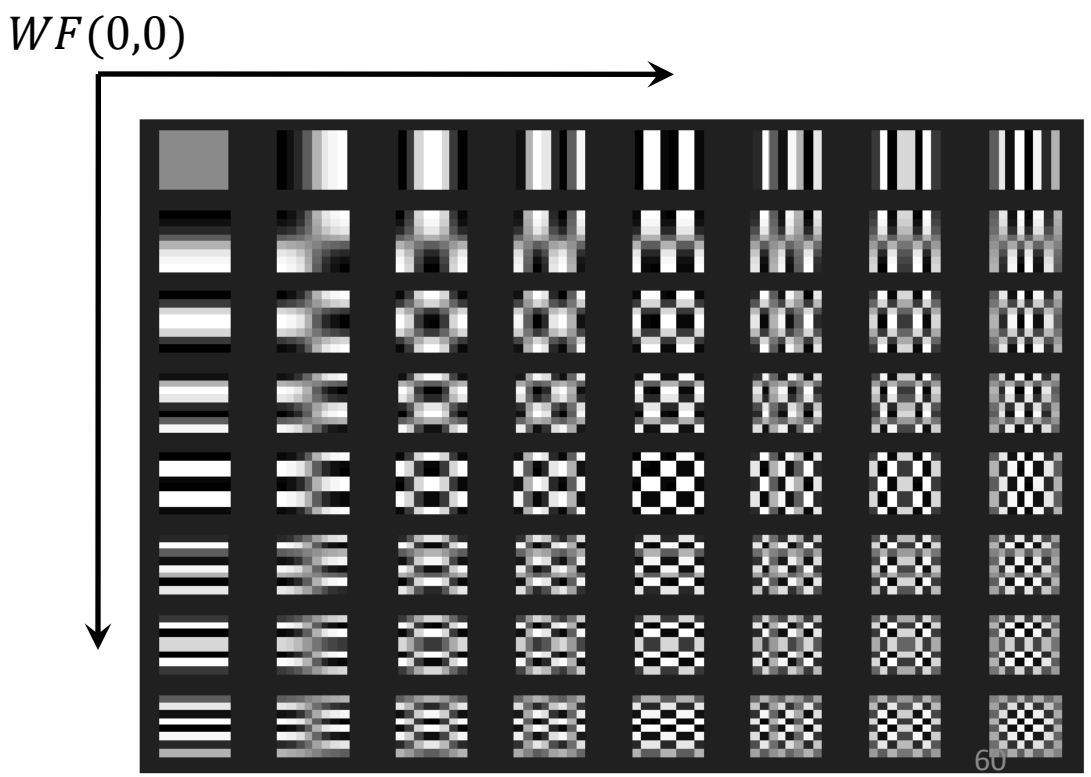
$$x(k,h) = \sum_{u=0}^{N-1}\sum_{v=0}^{M-1} \alpha(n)\alpha(m)X(n,m)\cos\left(\frac{\pi(2k+1)n}{2N}\right)\cos\left(\frac{\pi(2h+1)m}{2M}\right) \quad \begin{matrix} h = 0,1,2\ldots,N-1 \\ k = 0,1,2\ldots,M-1 \end{matrix}$$

- Multiplier:

$$\alpha(n) = \begin{cases} \sqrt{\frac{1}{N}} & n = 0 \\ \sqrt{\frac{2}{N}} & n \neq 0 \end{cases}$$

Discrete Cosine Transform (DCT)

- Basi ortogonali $WF(n, m)$ della 2-D DCT $(N = 8, M = 8)$
  - Grigio: zero
  - Nero: ampiezza negativa
  - Bianco: ampiezza positiva



Wavelet Compression

- Promettente alternativa alle classiche trasformate introdotte in precedenza
  - Può portare a migliori risultati in termini di compressione
- Le basi che compongono il segnale sono compressioni, dilatazioni e traslazioni di una funzione “madre” o *wavelet* (a differenza delle armoniche della DFT e delle cosine basis function della DCT)
- Vantaggi rispetto a DFT e DCT
  - Permette di rappresentare meglio segnali in cui vi sono discontinuità o bruschi cambiamenti (segnali non stazionari)
  - Si possono utilizzare diversi tipi di wavelet

Wavelet Decomposition

- Le funzioni che formano le basi nella analisi wavelet (monodimensionale) sono definite come

$$\phi_{s,k}(t) = 2^{-\frac{s}{2}}\phi(2^{-s}t - k)$$

$\phi(t)$ ⟵ **Analyzing Wavelet**

s ⟵ **Scale Index**

k ⟵ **Location Index**

- Gli indici s e k permettono di dilatare e traslare la Analyzing Wavelet

Wavelet Decomposition

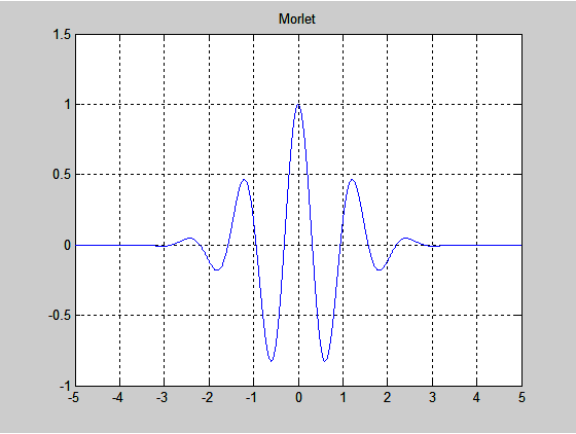
- Le funzioni base sono ottenute da compressioni/dilatazioni e traslazioni della *Analyzing Wavelet* e sono tra loro ortogonali
- Discrete Wavelet Transform (DWT)

$$d_{s,k} = \sum_{t=0}^{N-1} x(t)\phi_{s,k}(t)$$

Coefficienti della DWT

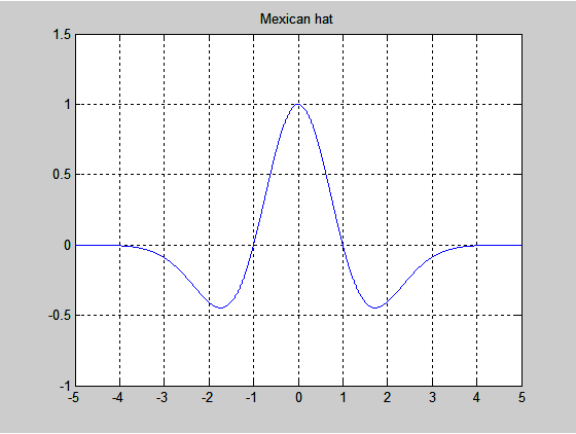
Esempi di Analyzing Wavelet

Morlet



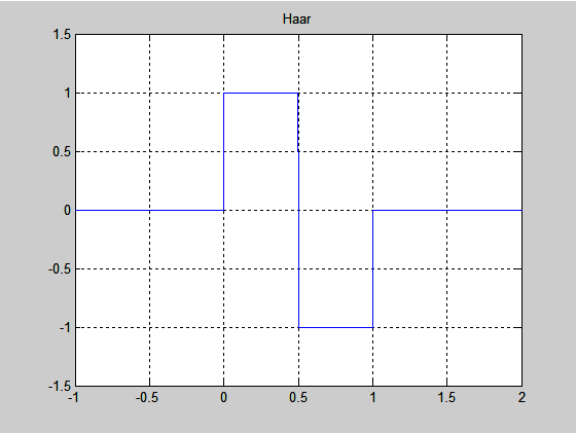
$$\phi(t) = e^{j\omega_0 t} e^{-\frac{t^2}{2}}$$

Mexican hat



$$\phi(t) = (1 - t^2) e^{-\frac{t^2}{2}}$$

Haar



$$\phi(t) = \begin{cases} 1 & 0 < t < \frac{1}{2} \\ -1 & \frac{1}{2} < t < 1 \\ 0 & altrimenti \end{cases}$$

Wavelet Decomposition 2-D

- La DWT 2-D è molto utile e utilizzata per la compressione di immagini
- Le tecniche di codifica wavelet sono di tipo progressivo e abilitano la *scalabilità*
  - Sfruttano la capacità della DWT di effettuare *un'analisi multi-risoluzione* dell'immagine
  - Aggiungendo coefficienti in fase di codifica di un immagine, è possibile migliorare progressivamente la qualità dell'immagine (proprietà utile nel caso di trasmissione dell'immagine)