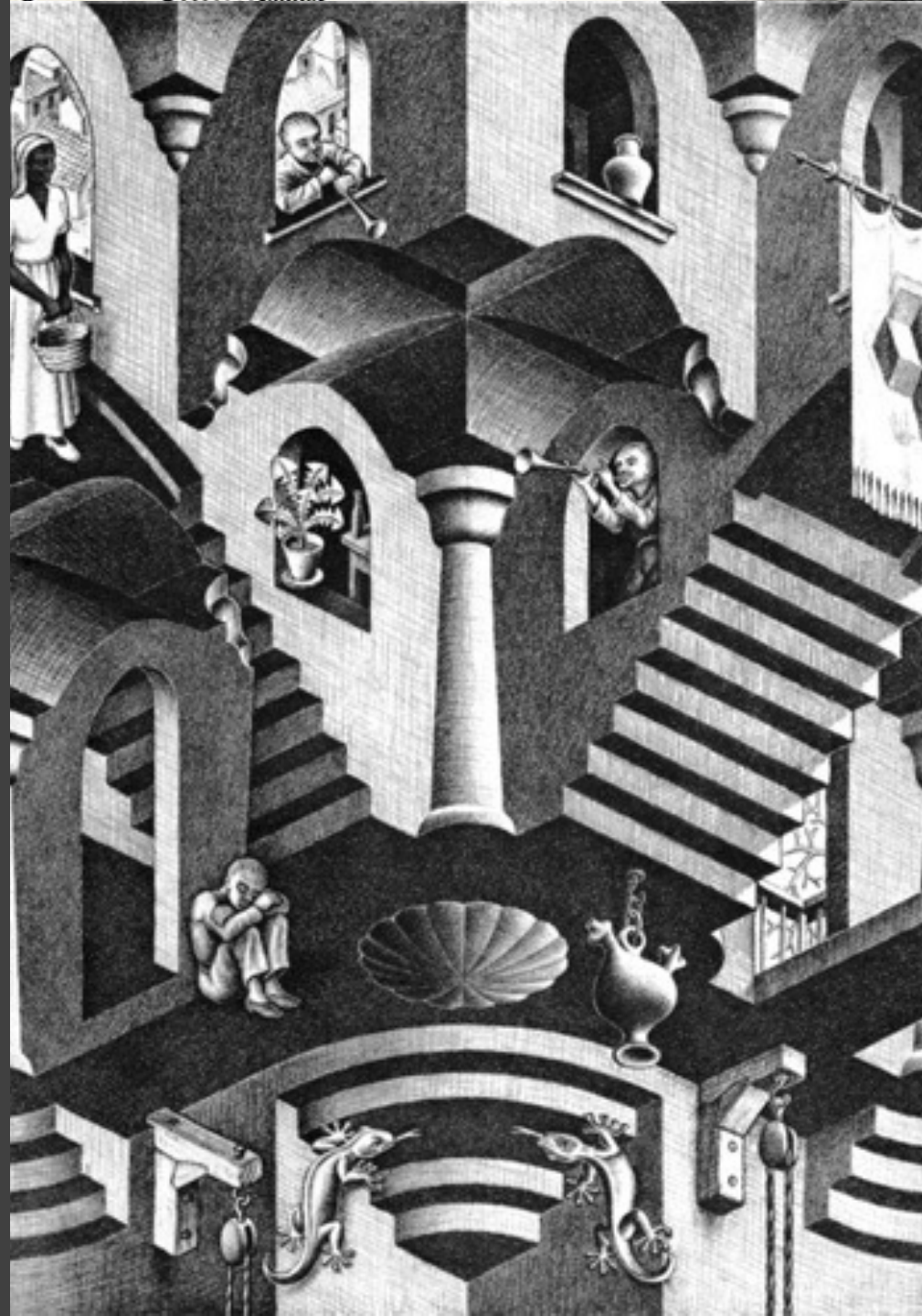
Università Roma Tre – Dipartimento di Matematica e Fisica
IN440 – Ottimizzazione Combinatoria
Prof. Marco Liverani

Codici di Huffman

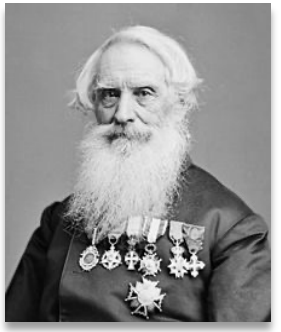


Codici

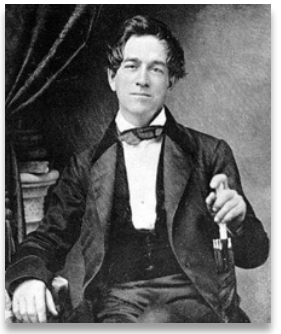
- Un **codice** è costituito da un *sistema* di simboli per rappresentare univocamente una determinata informazione; il **sistema** è composto da un **insieme di simboli** (**alfabeto**) e da un **insieme di sequenze** di simboli (**parole**) con cui si rappresenta (codifica) univocamente l'informazione

  Esempi:

  - il **codice ASCII** è un sistema con cui è possibile codificare i caratteri alfanumerici (alfabetici e numerici) utilizzando sequenze di 7 o di 8 bit (cifre binarie)
  - il **codice Morse** è un sistema con cui è possibile codificare i caratteri alfanumerici utilizzando sequenze di punti e linee (che possono essere fatti corrispondere rispettivamente a impulsi elettrici o segnali sonori di breve o di lunga durata)



Samuel Morse
(1791 – 1872)



Alfred Vail
(1807 – 1859)

Codice ASCII					
A	65	0100 0001	G	71	0100 0111
B	66	0100 0010	H	72	0100 1000
C	67	0100 0011	I	73	0100 1001
D	68	0100 0100	J	74	0100 1010
E	69	0100 0101	K	75	0100 1011
F	70	0100 0110	...		

Codice Morse			
A	• –	G	– – •
B	– • • •	H	• • • •
C	– • – •	I	• •
D	– • •	J	• – – –
E	•	K	– • –
F	• • – •	...	

Codici

- In generale quindi possiamo definire un codice come una corrispondenza tra gli elementi dell'alfabeto da codificare e sequenze di elementi dell'alfabeto del codice (codifiche)
- Una possibile classificazione dei codici è tra **codici a lunghezza fissa** e **codici a lunghezza variabile**
- **Codici a lunghezza fissa**: la codifica di ogni carattere dell'alfabeto da codificare avviene utilizzando sempre lo stesso numero di simboli dell'alfabeto di codifica
  - *Esempio*: il codice ASCII è un codice a lunghezza fissa: ogni carattere alfanumerico è codificato con un numero espresso in base 2 usando sempre 8 bit
  - È possibile decodificare un'informazione espressa con un codice a lunghezza fissa in tempo $O(n)$, scorrendo la sequenza da sinistra verso destra, senza dover utilizzare altri simboli in codice per rappresentare la terminazione di lettere o parole
- **Codici a lunghezza variabile**: la codifica di caratteri diversi dell'alfabeto da codificare può avvenire utilizzando un numero diverso di simboli dell'alfabeto di codifica
  - *Esempio*: il codice Morse è un codice a lunghezza variabile: la codifica della lettera «E» è data da un singolo punto «•», mentre la codifica del carattere «C» impiega quattro simboli «– • – •»
  - È più complicato decodificare il messaggio in codice, perché non possiamo suddividere la stringa a priori in sottostringhe di lunghezza prefissata corrispondenti alla codifica di singoli caratteri
  - Può essere però più efficiente come codifica, in termini di spazio di codifica o di tempo di trasmissione, perché è possibile codificare con sequenze più brevi i caratteri più frequenti (Morse studiò le frequenze dell'inglese)

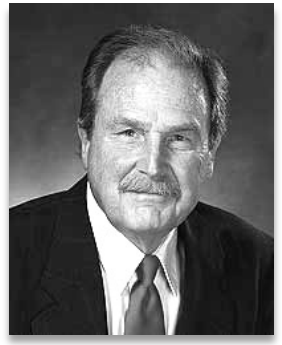
Codici prefissi

- Un messaggio in codice Morse, se non si utilizza un ulteriore simbolo come separatore dei caratteri codificati può generare delle ambiguità:

  C A N E ⟶ «– • – • | • – | – • | •» ⟶ «– • – • • – – • •» ⟶ «– • | – • | • | – – • | •» ⟶ N N E G E

- Per evitare ambiguità nella decodifica bisogna usare un terzo simbolo nell'alfabeto di codifica (nelle trasmissioni Morse si usa il tempo, per separare le lettere), oppure bisogna fare in modo che *nessun carattere abbia una codifica contenuta per intero all'inizio della codifica di un altro carattere*: il codice deve essere **prefisso**
- Nei codici prefissi il processo di decodifica è privo di ambiguità e può procedere da sinistra verso destra segmentando le sottostringhe non appena si forma la codifica di un carattere

Codici di Huffman



David Huffman
(1925 – 1999)

- Un **codice di Huffman** è un codice **binario**, **prefisso** a **lunghezza variabile**, basato su un algoritmo che adatta la codifica dei caratteri sulla base del testo da codificare, in modo da ottenere una **codifica ottima** che **minimizza** la dimensione della stringa codificata: si ottiene quindi la **compressione** del dato codificato
- Lo scopo di un codice di Huffman è di **produrre una codifica in formato compresso**
- Un codice di Huffman non è quindi una semplice corrispondenza «statica» tra caratteri e sequenze di simboli dell'alfabeto in codice, ma **è costituito da un vero e proprio algoritmo**, progettato con l'obiettivo di minimizzare il numero di cifre binarie per codificare un determinato testo
- David Huffman progettò questo algoritmo nel 1952 quando era studente di dottorato al MIT; oggi i codici di Huffman si ritrovano negli algoritmi dei programmi di compressione ZIP e nei «codec» per file multimediali come JPEG e MP3
- Il codice di Huffman per una determinata sequenza da comprimere viene definito costruendo un **albero binario** le cui foglie sono i caratteri da codificare e in cui, a partire dalla radice, ciascun cammino per raggiungere una foglia rappresenta una codifica binaria del carattere rappresentato dalla foglia, assumendo che «scendere verso il figlio sinistro» di un determinato vertice significhi inserire un bit 0 nella codifica del carattere e, viceversa, «scendere a destra» significhi aggiungere un bit 1 nella codifica

Codici di Huffman

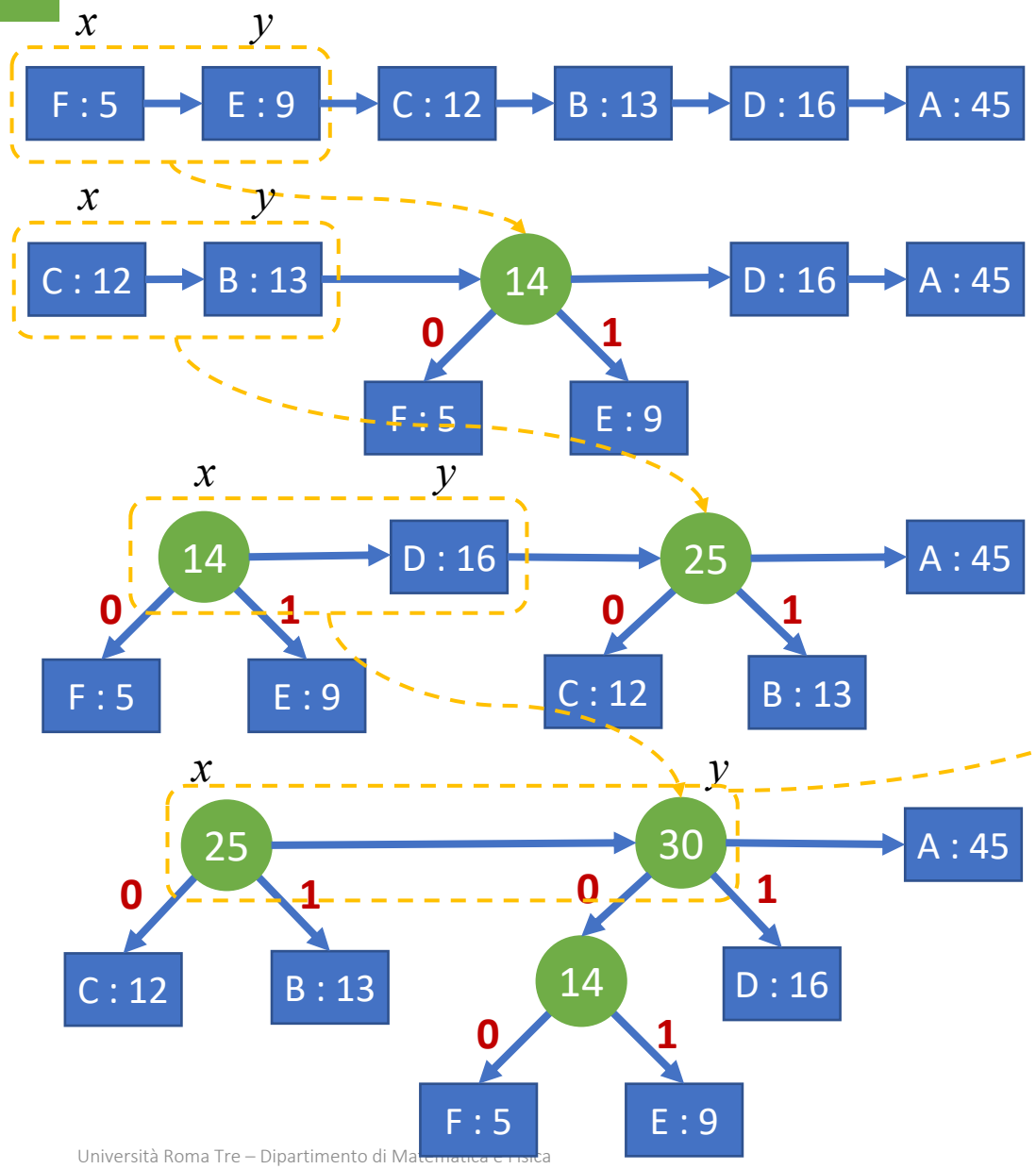
- La strategia è quella di costruire un albero binario in cui le foglie siano i caratteri da codificare e ogni cammino dalla radice ad una foglia sia la codifica del carattere corrispondente alla foglia stessa
- Per costruire l'albero binario con la codifica di Huffman si segue la seguente strategia:
  - vengono prima calcolate le frequenze dei caratteri nel testo originale da codificare e comprimere (il numero di occorrenze del carattere nel testo da comprimere)
  - quindi vengono elaborati i caratteri in ordine crescente di frequenza (prima i caratteri meno frequenti, poi quelli più frequenti), costruendo in modalità «bottom-up» l'albero binario
  - i dati vengono elaborati a coppie, sostituendo i due caratteri meno frequenti con un singolo «meta-carattere» che rappresenta un sotto-albero binario le cui foglie sono appunto i caratteri meno frequenti
  - Il processo quindi ad ogni iterazione sostituisce due caratteri o meta-caratteri con un solo meta-carattere, riducendo di uno la lunghezza della sequenza dei caratteri da codificare
  - il processo ha termine quando, dopo n iterazioni, la sequenza è ridotta ad un solo elemento, la radice dell'albero binario

Algoritmo 43 HUFFMAN(C)

Input: Un insieme $C = \{c_1, \ldots, c_n\}$ di caratteri con la frequenza $f(c_i)$
Output: Un albero binario con la codifica di Huffman per ogni carattere di C

1: $Q = C$
2: **per** $i = 1, \ldots, n-1$ **ripeti**
3: alloca un nuovo nodo z nell'albero binario
4: $z.left = x =$ EXTRACTMIN(Q)
5: $z.right = y =$ EXTRACTMIN(Q)
6: $z.freq = f(x) + f(y)$
7: inserisci z in Q
8: **fine-ciclo**

Codici di Huffman

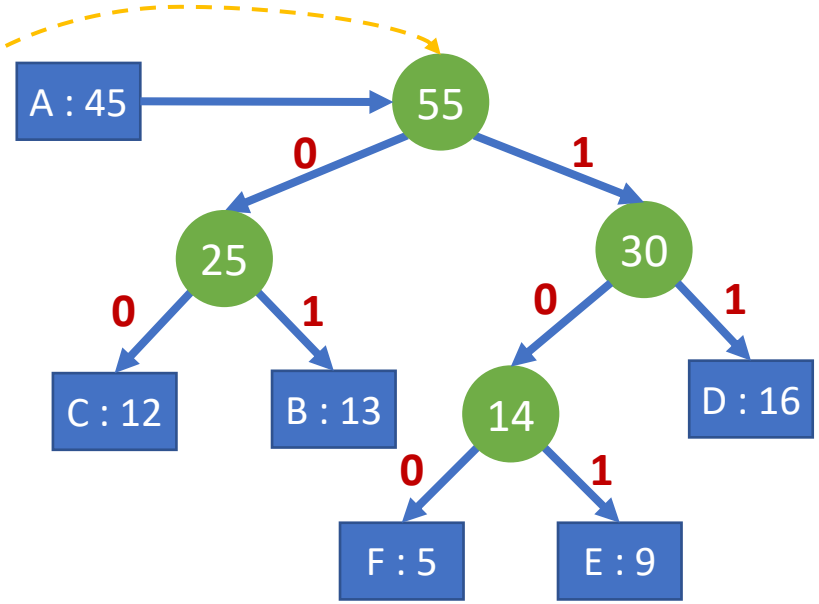


Algoritmo 43 HUFFMAN(C)

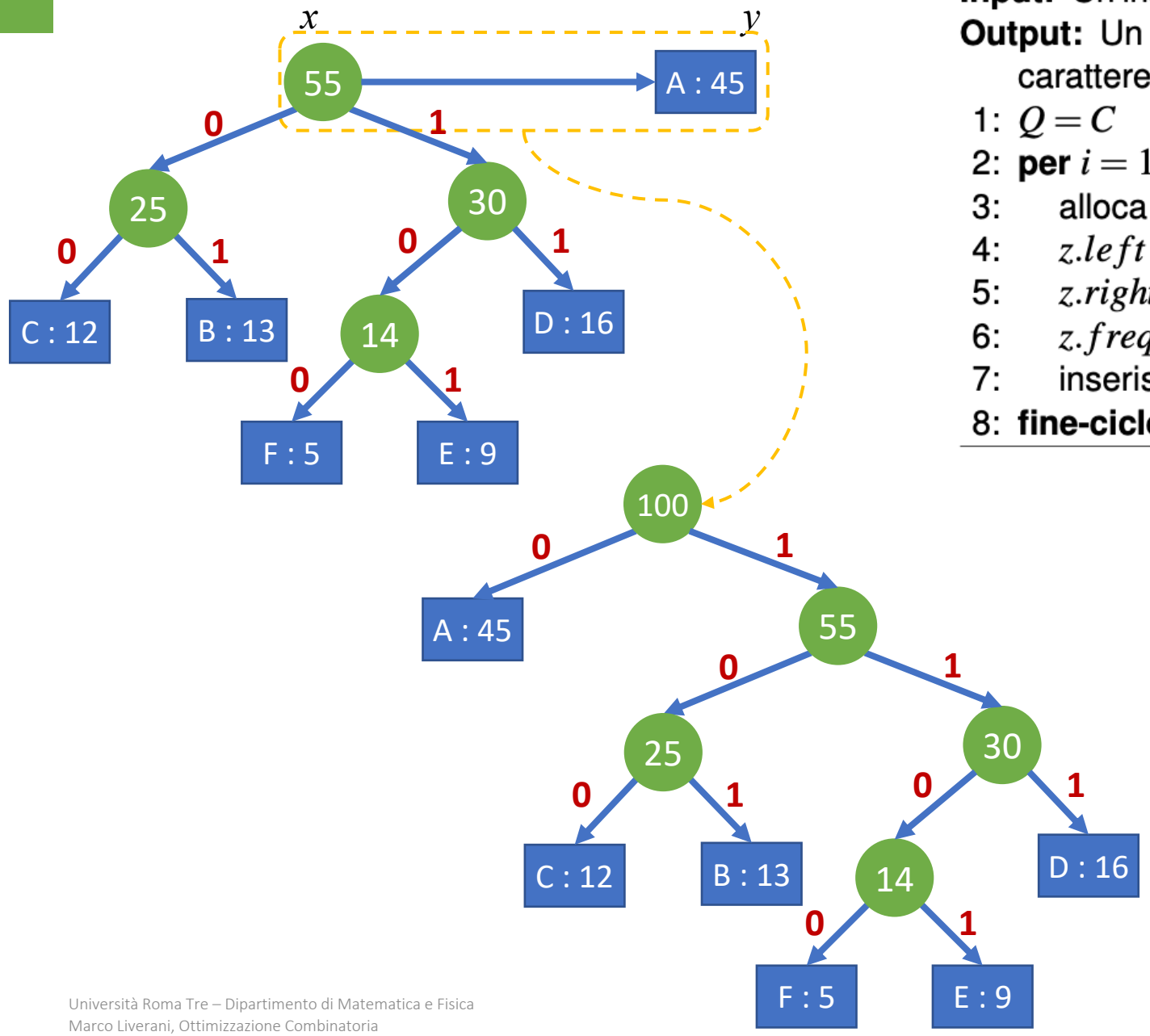
Input: Un insieme $C = \{c_1, \ldots, c_n\}$ di caratteri con la frequenza $f(c_i)$

Output: Un albero binario con la codifica di Huffman per ogni carattere di C

1: $Q = C$
2: **per** $i = 1, \ldots, n-1$ **ripeti**
3: alloca un nuovo nodo z nell'albero binario
4: $z.left = x =$ EXTRACTMIN(Q)
5: $z.right = y =$ EXTRACTMIN(Q)
6: $z.freq = f(x) + f(y)$
7: inserisci z in Q
8: **fine-ciclo**



Codici di Huffman



Algoritmo 43 HUFFMAN(C)

Input: Un insieme $C = \{c_1, \ldots, c_n\}$ di caratteri con la frequenza $f(c_i)$

Output: Un albero binario con la codifica di Huffman per ogni carattere di C

1: $Q = C$
2: **per** $i = 1, \ldots, n-1$ **ripeti**
3: alloca un nuovo nodo z nell'albero binario
4: $z.left = x =$ EXTRACTMIN(Q)
5: $z.right = y =$ EXTRACTMIN(Q)
6: $z.freq = f(x) + f(y)$
7: inserisci z in Q
8: **fine-ciclo**

$O(n \log_2 n)$

$O(\log_2 n)$ se Q è un heap binario

Huffman	
A (45):	0
D (16):	1 1 1
B (13):	1 0 1
C (12):	1 0 0
E (9):	1 1 0 1
F (5):	1 1 0 0

$45 \times 1 + 3 \times 3 + 13 \times 3 + 12 \times 3 + 9 \times 4 + 5 \times 4 = 224$

Fixed Length	
A (45):	0 0 1
D (16):	0 1 0
B (13):	0 1 1
C (12):	1 0 0
E (9):	1 0 1
F (5):	1 1 1

$45 \times 3 + 3 \times 3 + 13 \times 3 + 12 \times 3 + 9 \times 3 + 5 \times 3 = 300$

Riferimenti bibliografici

- Cormen, Leiserson, Rivest, Stein, «*Introduzione agli algoritmi e strutture dati*», terza edizione, McGraw-Hill (Cap. 16.3)