Codici prefissi

Un ***codice prefisso*** è un codice in cui nessuna parola codice è prefisso (parte iniziale) di un'altra

Ogni codice a lunghezza fissa è ovviamente prefisso.

Ma anche il codice a lunghezza variabile che abbiamo appena visto è un codice prefisso.

Codifica e decodifica sono semplici con i codici prefissi.

Con il codice prefisso

carattere	a	b	c	d	e	f
cod. var.	0	101	100	111	1101	1100

la codifica della stringa **abc** è

0101100

La decodifica è pure semplice.

Siccome nessuna parola codice è prefisso di un'altra, la prima parola codice del file codificato risulta univocamente determinata.

Per la decodifica basta quindi:

1. individuare la prima parola codice del file codificato
2. tradurla nel carattere originale e aggiungere tale carattere al file decodificato
3. rimuovere la parola codice dal file codificato
4. ripetere l’operazione per i caratteri successivi

Ad esempio con il codice

carattere	a	b	c	d	e	f
cod. var.	0	101	100	111	1101	1100

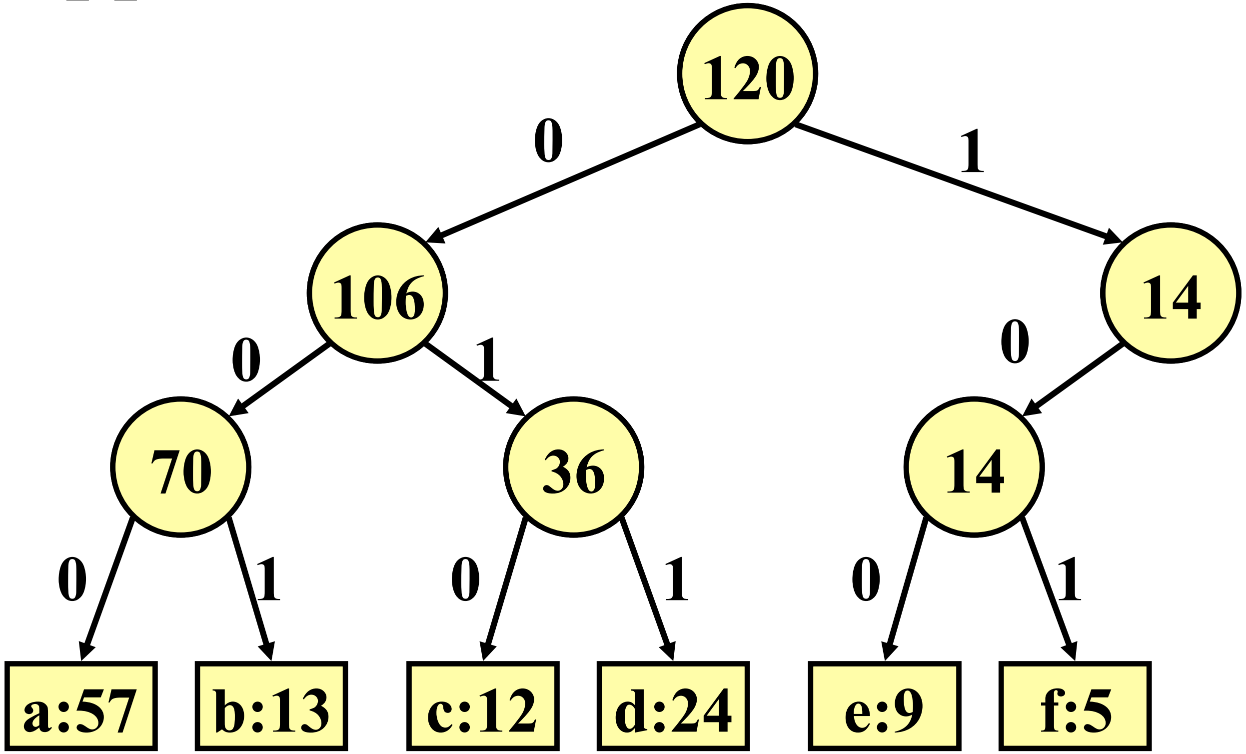
la suddivisione in parole codice della stringa di bit **001011101** è **0·0·101·1101** a cui corrisponde la stringa **aabe**

Per facilitare la suddivisione del file codificato in parole codice è comodo rappresentare il codice con un albero binario.

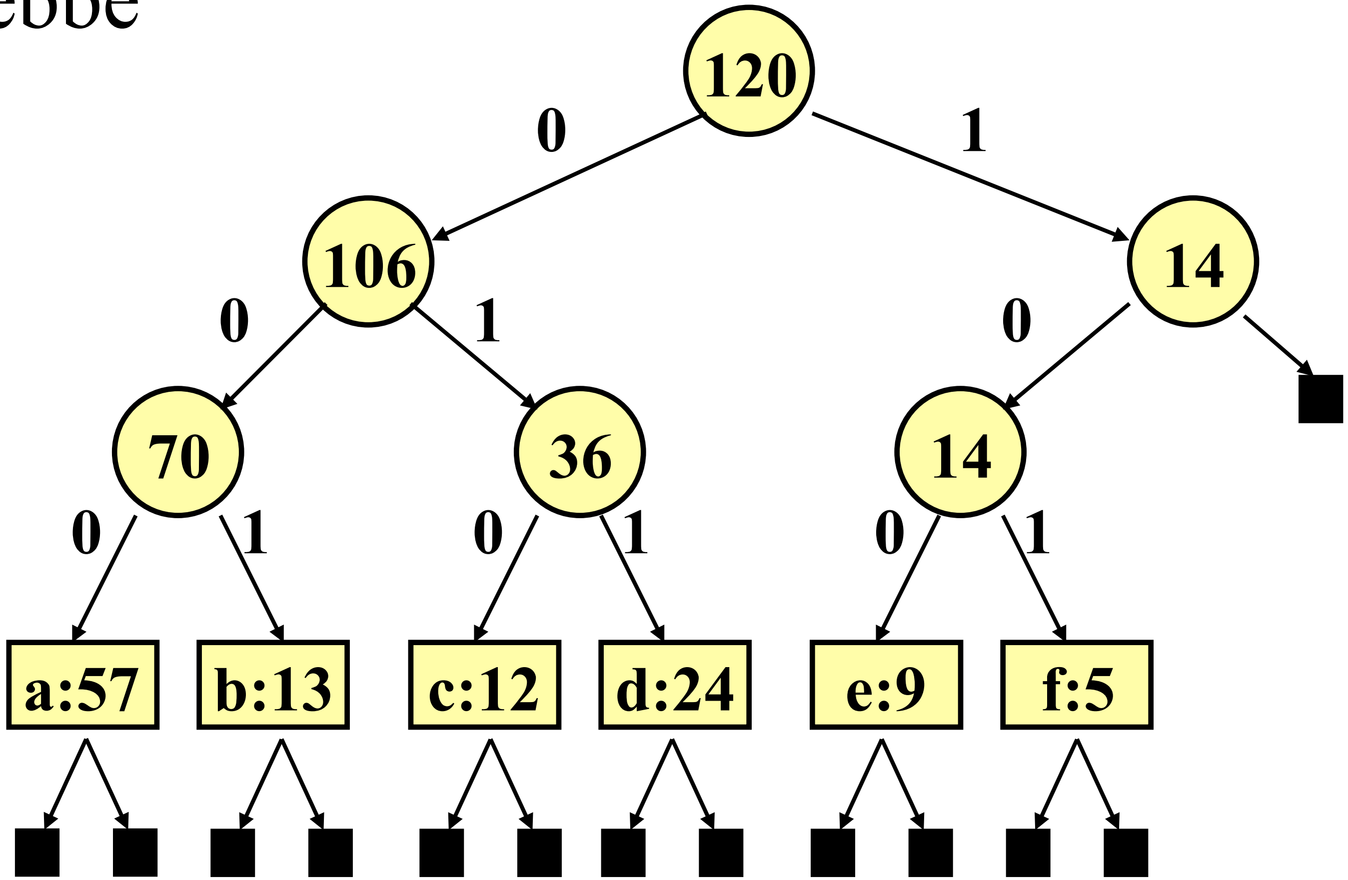
Esempio: il codice a lunghezza fissa

carattere	a	b	c	d	e	f
frequenza	57	13	12	24	9	5
cod. fisso	000	001	010	011	100	101

ha la rappresentazione ad albero



In realtà, come albero binario, la rappresentazione sarebbe

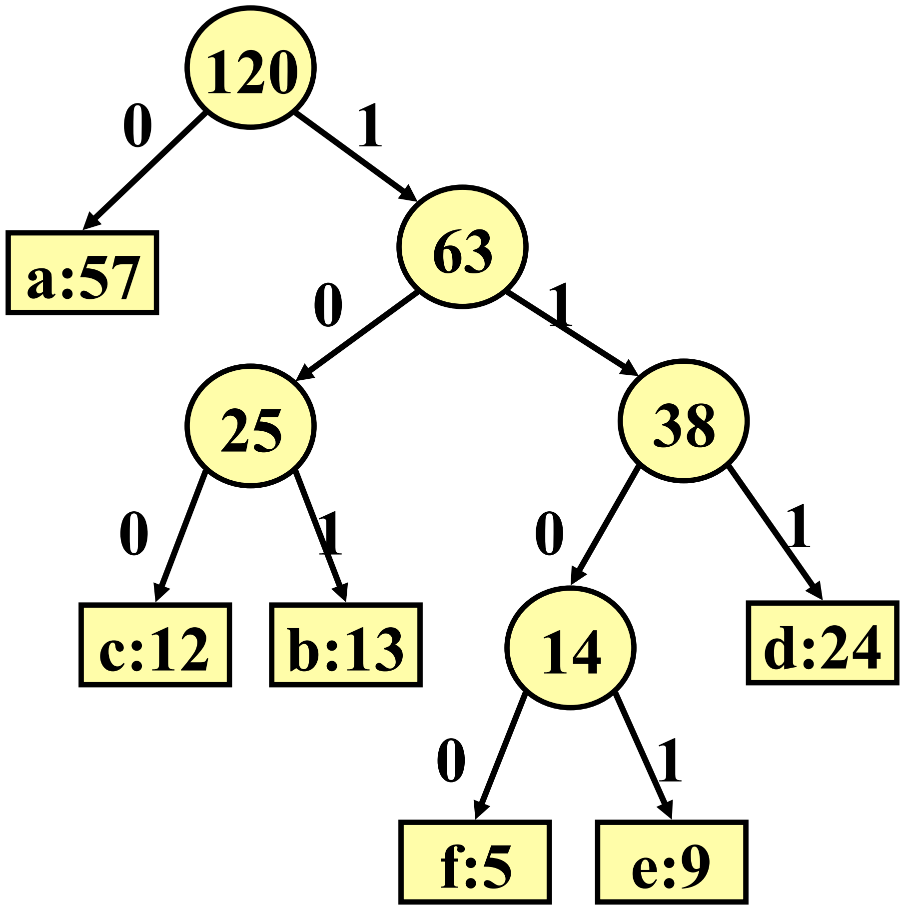


Noi eliminiamo le foglie e chiamiamo foglie i nodi interni senza figli

Il codice a lunghezza variabile

carattere	**a**	**b**	**c**	**d**	**e**	**f**
frequenza	**57**	**13**	**12**	**24**	**9**	**5**
cod. var.	**0**	**101**	**100**	**111**	**1101**	**1100**

è rappresentato



La lunghezza in bit del file codificato con il codice rappresentato da un albero T è:

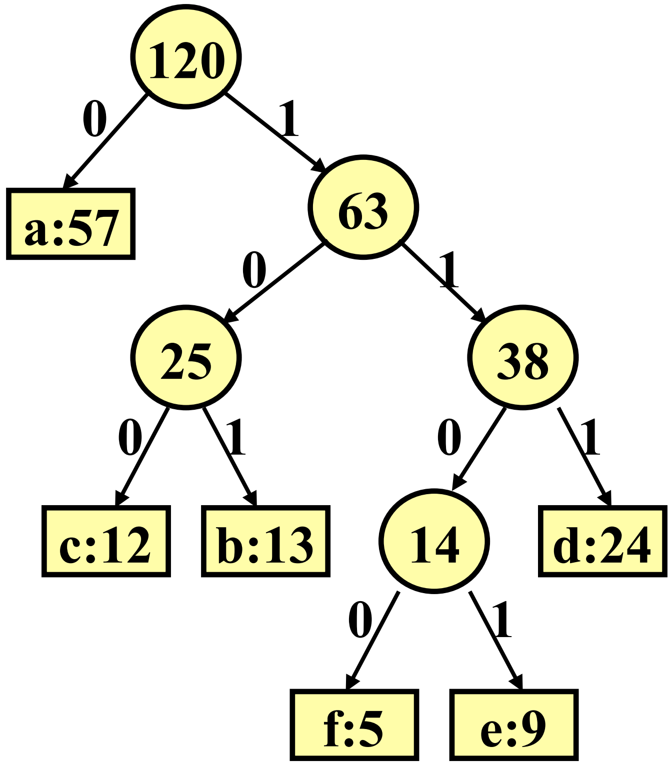
$$B(T) = \sum_{c \in \Sigma} f_c d_T(c)$$

in cui la sommatoria è estesa a tutti i caratteri c dell'alfabeto Σ, f_c è la frequenza del carattere c e $d_T(c)$ è la profondità della foglia che rappresenta il carattere c nell'albero T

Nota: assumiamo che l'alfabeto Σ contenga almeno due caratteri. In caso contrario basta un numero per rappresentare il file: la sua lunghezza

La lunghezza in bit del file codificato è anche:

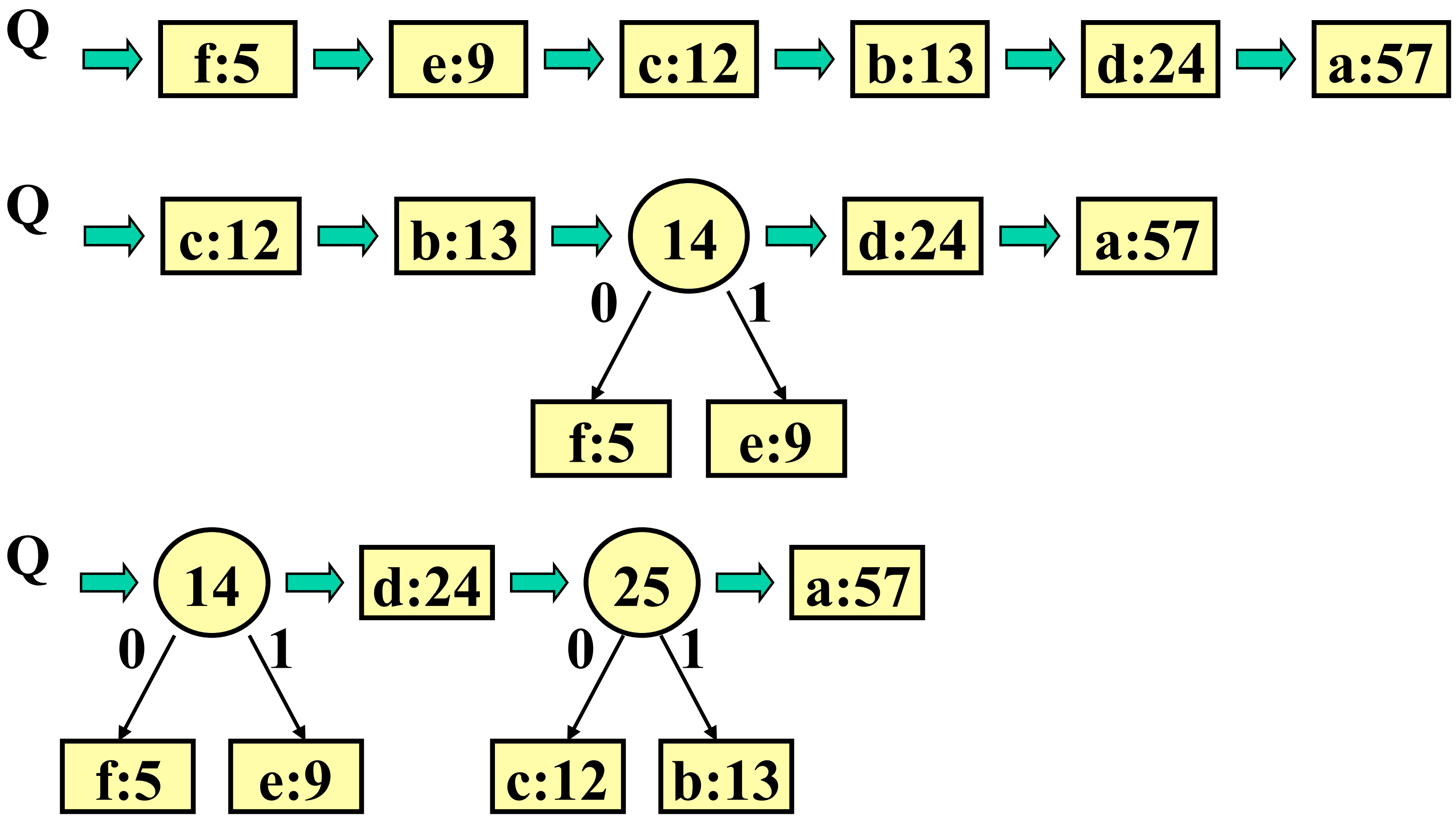
$$B(T) = \sum_{x \text{ nodo interno}} x.f$$

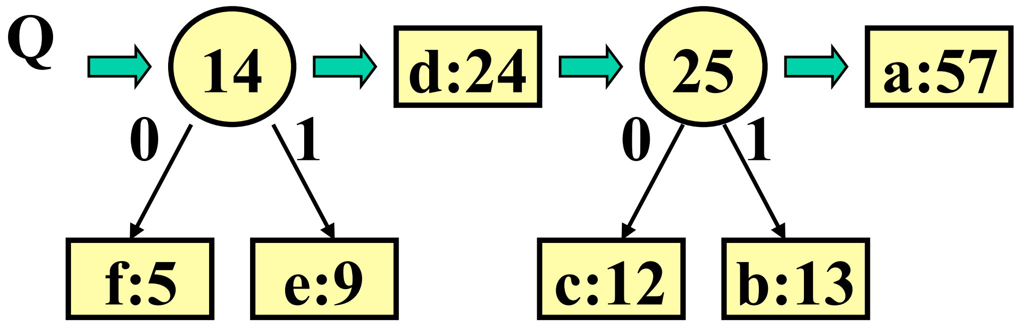


in cui la sommatoria è estesa alle frequenze ***x.f*** di tutti i nodi interni ***x*** dell'albero ***T.***

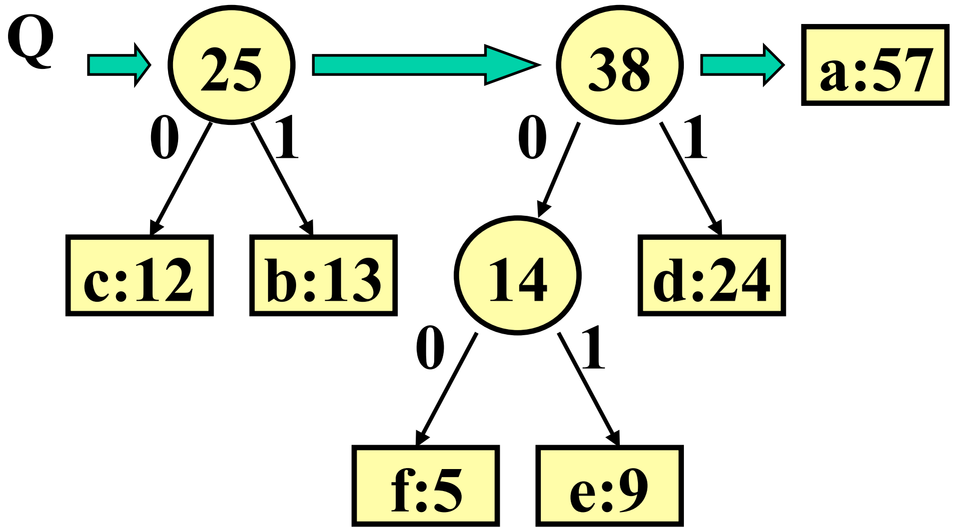
Costruzione dell’albero di Huffman:

carattere	a	b	c	d	e	f
frequenza	57	13	12	24	9	5

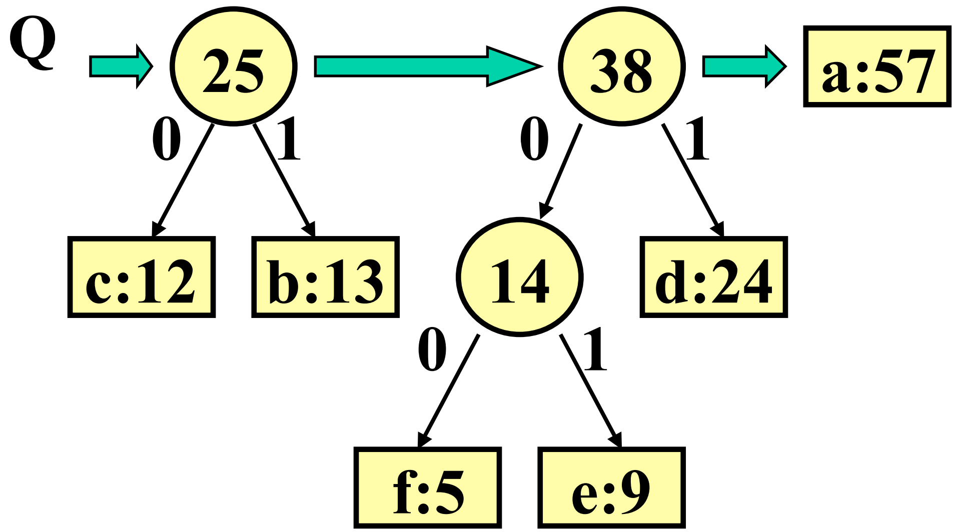


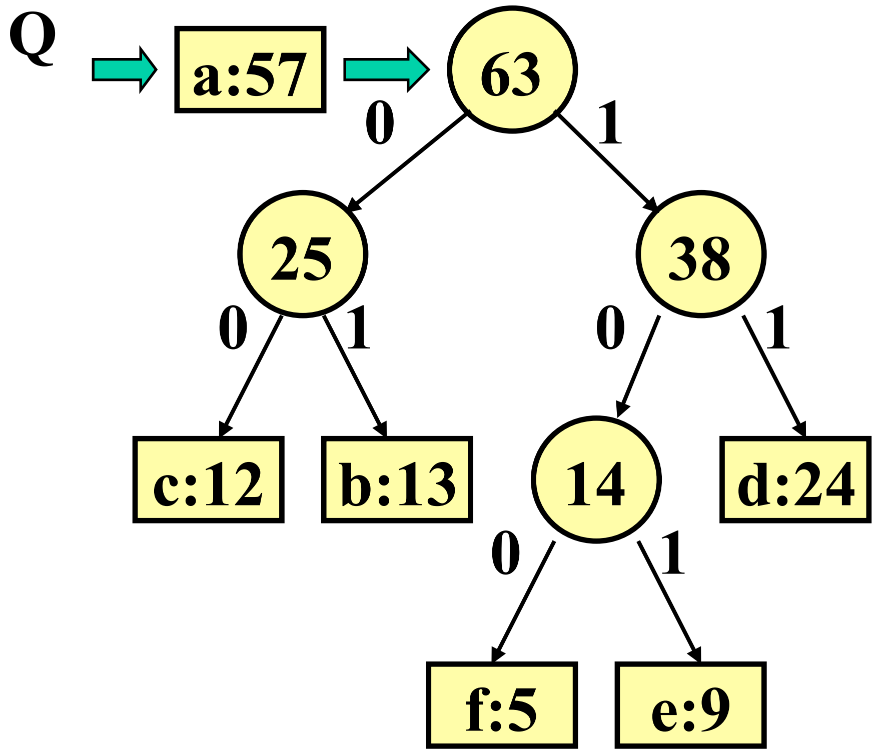


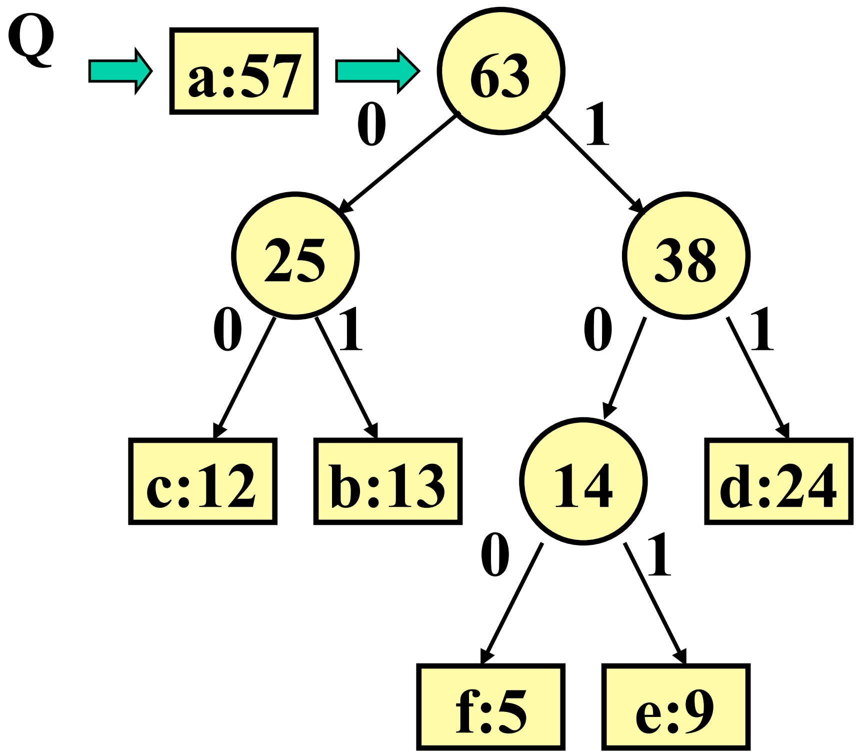
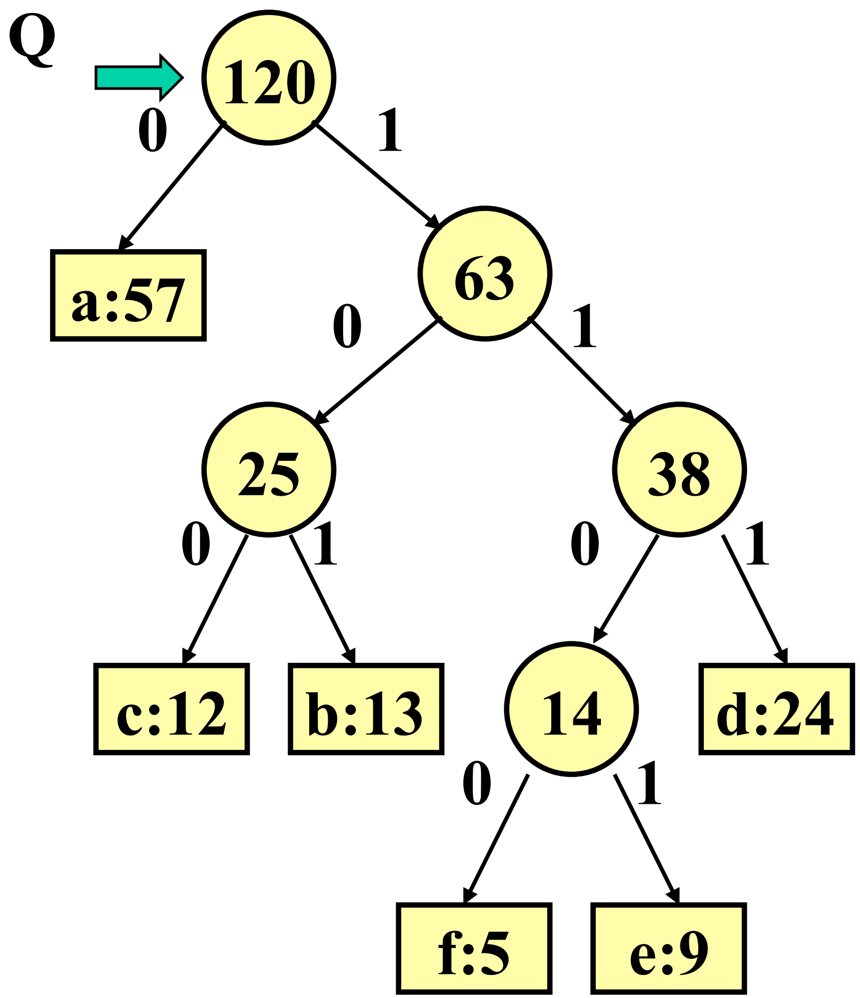
Q
14
d:24
25
a:57
0
1
0
1
f:5
e:9
c:12
b:13



Q
25
38
a:57
0
1
0
1
c:12
b:13
14
d:24
0
1
f:5
e:9


Q
25
38
a:57
0
1
c:12
b:13
14
d:24
f:5
e:9


Q
a:57
63
0
1
25
38
c:12
b:13
14
d:24
f:5
e:9


Q
a:57
63
25
38
c:12
b:13
14
d:24
f:5
e:9
0
1

Q
120
a:57
63
25
38
c:12
b:13
14
d:24
f:5
e:9
0
1

Implementazione dell'algoritmo goloso di Huffman

```
Huffman(c, f, n)
   Q = Ø                          // coda con priorità
   for i = 1 to n
      Push(Q, Nodo(f_i, c_i))
   for j = n downto 2
      x = ExtractMin(Q)
      y = ExtractMin(Q)
      Push(Q, Nodo(x, y))
   return ExtractMin(Q)
```

Nodo(f, c) costruttore dei nodi foglia

Nodo(x, y) costruttore dei nodi interni

Assumendo che la coda ***Q*** venga realizzata con un heap, le operazioni ***Insert*** ed ***ExtractMin*** richiedono tempo $\boldsymbol{O(\log n)}$.

Pertanto l'algoritmo richiede tempo $\boldsymbol{O(n \log n)}$ (dove ***n*** è il numero di caratteri dell'alfabeto).

L’algoritmo è ***goloso*** perché ad ogni passo costruisce il nodo interno avente frequenza minima possibile.

Ricordiamo infatti che

$$\boldsymbol{B(T)} = \sum_{x \text{ nodo interno}} \boldsymbol{x.f}$$

Siamo sicuri che in questo modo otteniamo sempre un codice ottimo?

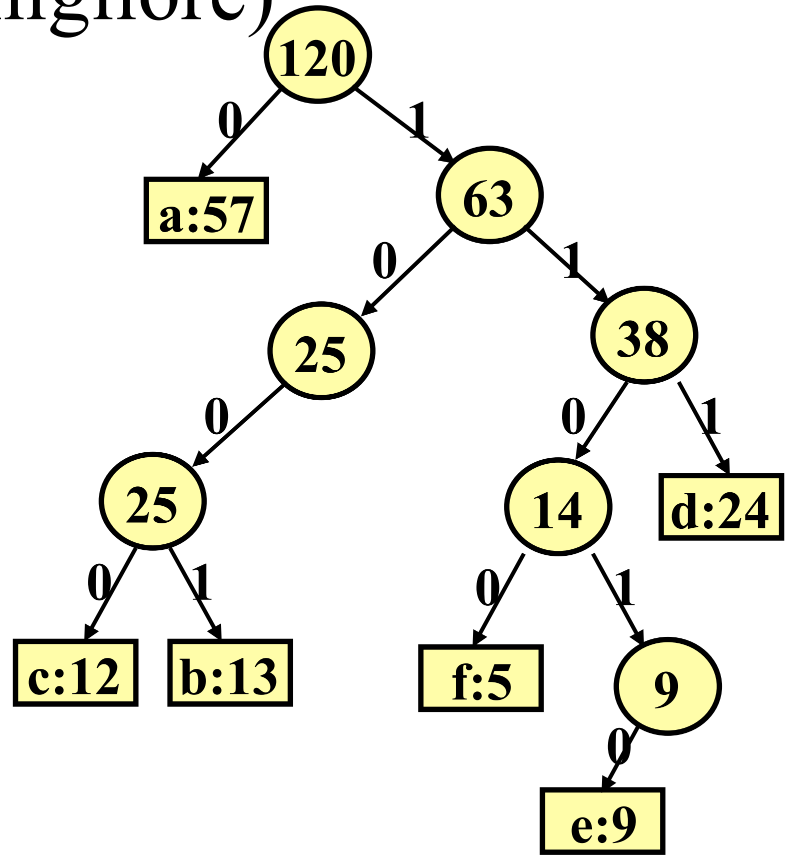
Elementi della strategia golosa

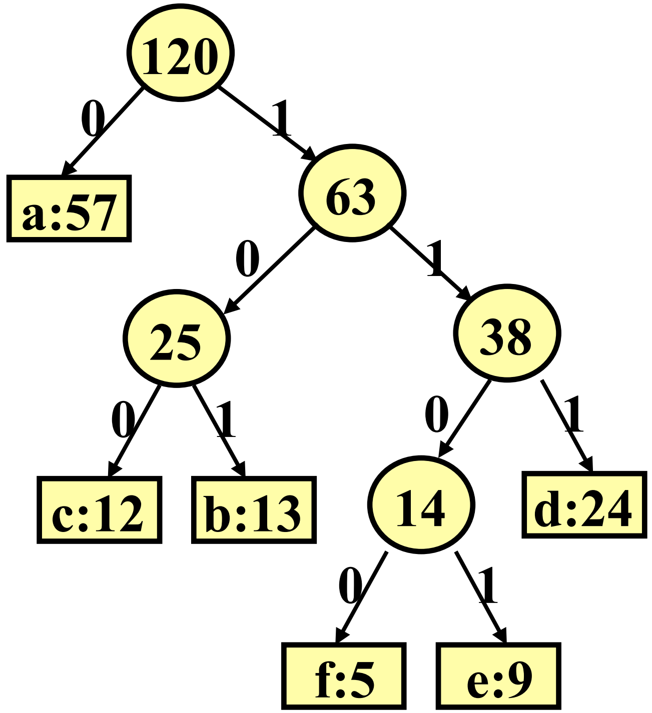
Ingredienti comuni a molti problemi risolvibili con la strategia golosa:

Sottostruttura ottima: Ogni soluzione ottima non elementare si ottiene da soluzioni ottime di sottoproblemi.

Proprietà della scelta golosa: La scelta localmente ottima (golosa) non pregiudica la possibilità di arrivare ad una soluzione globalmente ottima.

Se T è ottimo ogni nodo interno ha due figli (altrimenti togliendo il nodo si otterrebbe un codice migliore)





Se T è ottimo esistono due foglie sorelle x ed y a profondità massima.

Proprietà (*sottostruttura ottima*)

Sia $\boldsymbol{T}$ l'albero di un codice prefisso ottimo per l'alfabeto $\boldsymbol{\Sigma}$ e siano $\boldsymbol{a}$ ed $\boldsymbol{b}$ i caratteri associati a due foglie sorelle $\boldsymbol{x}$ ed $\boldsymbol{y}$ di $\boldsymbol{T}$.

Se consideriamo il padre $\boldsymbol{z}$ di $\boldsymbol{x}$ ed $\boldsymbol{y}$ come foglia associata ad un nuovo carattere $\boldsymbol{c}$ con frequenza

$$\boldsymbol{f_c = z.f = f_a + f_b}$$

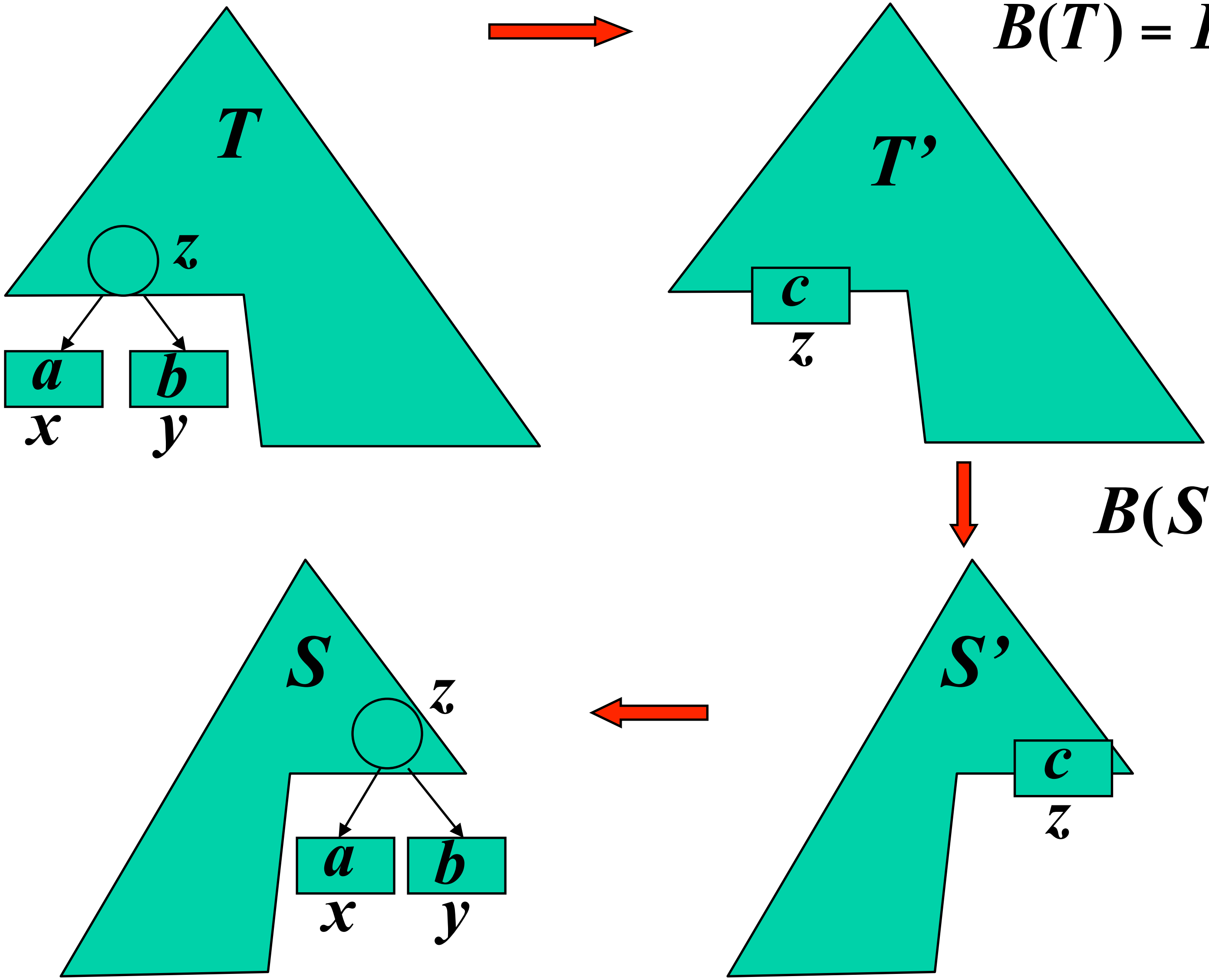
allora l'albero $\boldsymbol{T' = T - \{x,y\}}$ rappresenta un codice prefisso ottimo per l'alfabeto

$$\boldsymbol{\Sigma' = \Sigma - \{a,b\} \cup \{c\}}$$

$$\begin{aligned} B(T) &= B(T') + f_a d_T(a) + f_b d_T(b) - f_c d_{T'}(c) \\ &= B(T') + (f_a + f_b)(d_{T'}(c) + 1) - (f_a + f_b) d_{T'}(c) \\ &= B(T') + f_a + f_b \end{aligned}$$

Supponiamo, per assurdo, esista un albero S' per Σ' tale che $B(S') < B(T')$.

Aggiungendo ad S' le foglie x ed y come figlie del nodo z (che in S' è una foglia) otterremmo un albero S per Σ tale che $B(S) < B(T)$ contro l'ipotesi che T sia ottimo.



$$B(T) = B(T') + f_a + f_b$$

$$B(S') < B(T')$$

$$B(S) = B(S') + f_a + f_b < B(T') + f_a + f_b = B(T)$$

Proprietà (*scelta golosa*)

Siano ***a*** e ***b*** due caratteri di Σ aventi frequenze f_a ed f_b minime

Esiste un codice prefisso ottimo in cui le parole codice di ***a*** e ***b*** hanno uguale lunghezza e differiscono soltanto per l'ultimo bit.

Se i codici di ***a*** e ***b*** differiscono soltanto per l'ultimo bit, nell'albero del codice le foglie ***a*** e ***b*** sono figlie dello stesso nodo, cioè sorelle.

Attenzione: Il Lemma ***non*** dice che ciò è vero per ogni codice prefisso ottimo e ***tanto meno*** che se ciò è vero il codice è ottimo!!!!

Dice ***solo*** che ciò è vero per ***almeno*** un codice ottimo.

Sappiamo che in $\boldsymbol{T}$ esistono due foglie sorelle a profondità massima.

Siano $\boldsymbol{c}$ e $\boldsymbol{d}$ i caratteri di tali foglie.

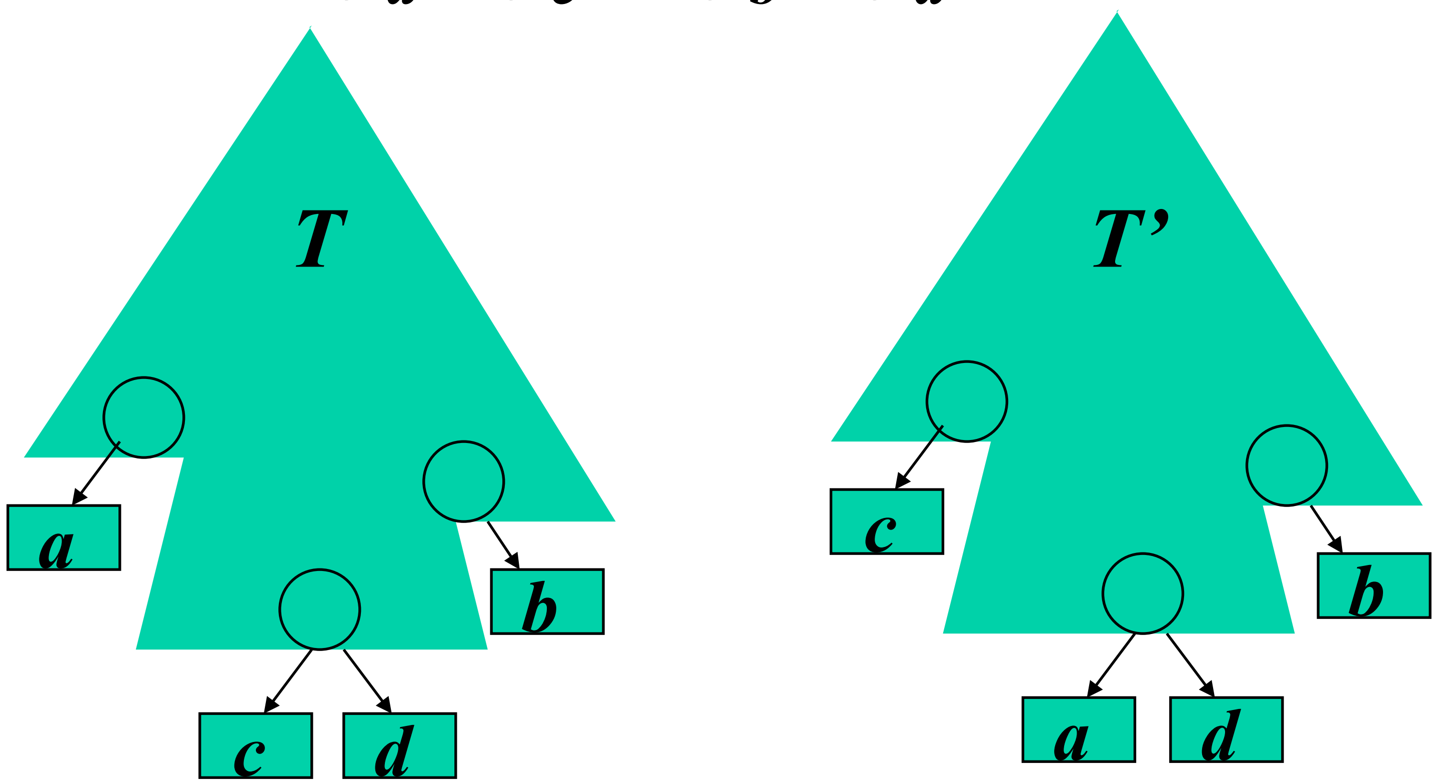
Mostriamo che scambiando $\boldsymbol{c}$ e $\boldsymbol{d}$ con $\boldsymbol{a}$ e $\boldsymbol{b}$ il codice rimane ottimo.

Possiamo supporre $\boldsymbol{f_c \leq f_d}$ ed $\boldsymbol{f_a \leq f_b}$.

$\boldsymbol{a}$ e $\boldsymbol{b}$ sono i due caratteri con frequenza minima in assoluto e quindi $\boldsymbol{f_a \leq f_c}$ ed $\boldsymbol{f_b \leq f_d}$.

Sia ***T'*** ottenuto da ***T*** scambiando la foglia ***c*** con la foglia ***a*** (con ricalcolo delle frequenze dei nodi interni)

$$f_a \le f_c \text{ ed } f_b \le f_d.$$



Allora:

$$
\begin{aligned}
B(T) - B(T') &= \sum_{k\in\Sigma} f_k d_T(k) - \sum_{k\in\Sigma} f_k d_{T'}(k) \\
&= f_a d_T(a) + f_c d_T(c) - f_a d_{T'}(a) - f_c d_{T'}(c) \\
&= f_a d_T(a) + f_c d_T(c) - f_a d_T(c) - f_c d_T(a) \\
&= [f_c - f_a][d_T(c) - d_T(a)] \\
&\geq 0
\end{aligned}
$$

Siccome T è ottimo $B(T) = B(T')$ e quindi anche T' è ottimo.

Scambiando poi le foglie d e b, si ottiene ancora un albero ottimo T'' in cui a e b sono foglie sorelle.

Teorema L'algoritmo di Huffman produce un codice prefisso ottimo

```
Huffman(c, f, n)
    Q = Ø                           // coda con priorità
    for i = 1 to n
        Push(Q, Nodo(f_i, c_i))
    for j = n downto 2
        x = ExtractMin(Q)
        y = ExtractMin(Q)
        Push(Q , Nodo(x,y))
    return ExtractMin(Q)
```

proprietà della scelta golosa

Esercizio 5

Dimostrare che ogni algoritmo di compressione che accorcia qualche sequenza di bit deve per forza allungarne qualche altra.

Dimostrazione

Supponiamo per assurdo che l'algoritmo accorci qualche sequenza ma non ne allunghi nessuna.

Sia $\boldsymbol{x}$ la più corta sequenza che viene accorciata dall'algoritmo e sia $\boldsymbol{m}$ la sua lunghezza.

Le sequenze di lunghezza minore di $\boldsymbol{m}$ sono $\mathbf{2^{\boldsymbol{m}}-1}$ e non vengono accorciate o allungate.

Ognuna di esse viene codificata con una sequenza diversa e quindi le loro codifiche sono anch'esse $\mathbf{2^{\mathit{m}}-1}$ e sono tutte di lunghezza minore di ***m*.**

Dunque ognuna delle $\mathbf{2^{\mathit{m}}-1}$ sequenze più corte di ***m*** è codifica di qualche sequenza più corta di ***m*.**

Dunque la codifica di ***x*** coincide con la codifica di un'altra sequenza. ASSURDO!!!!