MASTER THEOREM - RICORRENZE

$[n, k] \rightarrow$ TROVI QUESTE

$$3\frac{n}{2}\left(\frac{n}{2}+1\right) < kn(n+1)$$

$$\frac{3n^2}{4} + \frac{3n}{2} < kn^2 + 1$$

$\Theta$ $O$ $\Omega$

$$4\,\frac{3n^2+6n}{\cancel{4}} < 4kn^2+4$$

$$3n^2 + 6n < 4kn^2 + 4 \Rightarrow a\,f\left(\frac{n}{b}\right) < k\,f(n)$$

$$\sim \quad 3n^2 < 4kn^2$$

$$\left[k > \frac{4}{3}, \forall n \in \mathbb{N}\right] \qquad T(n) = \Theta(f(n)) \quad O \quad \Omega$$

↑ 1 k, 1 N

✓ ESERCIZI "BREVI"

**Domanda B** (6 punti) Si calcoli la lunghezza della longest common subsequence (LCS) tra le stringhe *armo* e *toro*, calcolando tutta la tabella $L[i, j]$ delle lunghezze delle LCS sui prefissi usando l'algoritmo visto in classe.

**Soluzione:** Si ottiene

| | | $t$ | $o$ | $r$ | $o$ |
|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 |
| $a$ | 0 | 0 | 0 | 0 | 0 |
| $r$ | 0 | 0 | 0 | 1 | 1 |
| $m$ | 0 | 0 | 0 | 1 | 1 |
| $o$ | 0 | 0 | 1 | 1 | 2 |

La lunghezza della longest common subsequence tra *armo* e *toro* è quindi 2.

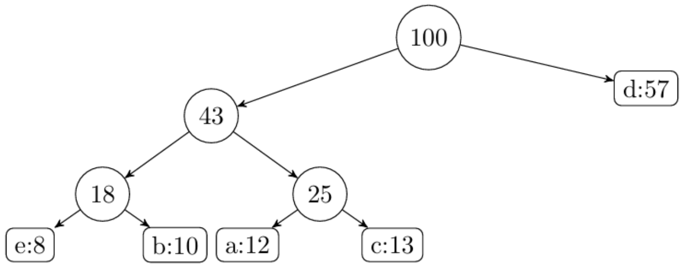
**Domanda B** (5 punti) Indicare, in forma di albero binario, il codice prefisso ottenuto tramite l'algoritmo di Huffmann per l'alfabeto $\{a, b, c, d, e\}$, supponendo che ogni simbolo appaia con le seguenti frequenze:

| a | b | c | d | e |
|---|---|---|---|---|
| 12 | 10 | 13 | 57 | 8 |

← FREQ. PIÙ BASSE SCEGLIE

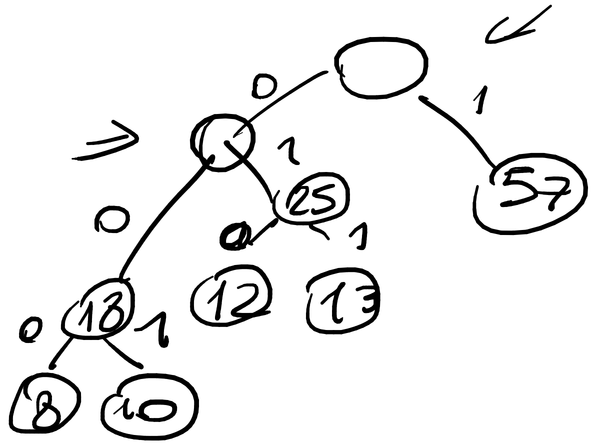
Spiegare il processo di costruzione del codice.

**Soluzione:** Per la descrizione del processo di costruzione, si rimanda al libro. Il risultato è il seguente:



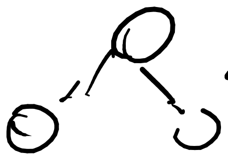

NODO SOLO = OTTIMO

CODIFICA NON LIBERA DA PREFISSI




[010] π

[01] (!)



← OGNI NODO DOVE AVERE UN NODO SOMMA

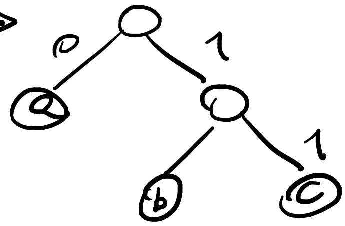
ESEMPIO NOTEVOLE (19-20)

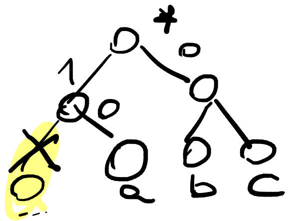
**Esercizio 2** (9 punti) Si consideri un file definito sull'alfabeto $\Sigma = \{a, b, c\}$, con frequenze $f(a), f(b), f(c)$. Per ognuna delle seguenti codifiche si determini, se esiste, un opportuno assegnamento di valori alle 3 frequenze $f(a), f(b), f(c)$ per cui l'algoritmo di Huffman restituisce tale codifica, oppure si argomenti che tale codifica non è mai ottenibile.

SI → 1. $e(a) = 0,\ e(b) = 10,\ e(c) = 11$

NO → 2. $e(a) = 1,\ e(b) = 0,\ e(c) = 11$

3. $e(a) = 10,\ e(b) = 01,\ e(c) = 00$ NO

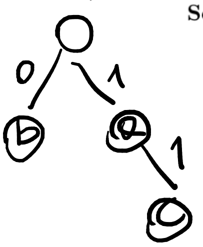




NON COMPLETO → NO

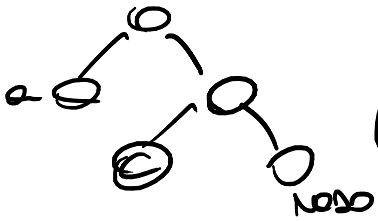
CAMMINO BISOGNO SI ×

**Soluzione:**

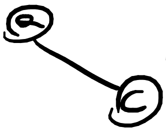


1. Questa codifica viene restituita dall'algoritmo di Huffman quando $f(b), f(c) < f(a)$: in questo caso i nodi associati a $b$ e $c$ vengono uniti creando un nuovo nodo interno, che poi viene unito al nodo associato ad $a$. Quindi, per esempio, $f(a) = 40, f(b) = 25, f(c) = 35$.
2. La codeword $e(a) = 1$ è un prefisso della codeword $e(c) = 11$, cioè questa codifica non è libera da prefissi, e quindi non è una codifica di Huffman.
3. L'albero associato a questa codifica non è pieno perché c'è un nodo interno con un solo figlio (quello nel cammino che porta alla foglia associata al carattere $a$). Quindi questa codifica non è ottima e quindi non è una codifica di Huffman.

PREFISSI



(!) SI → 



(!!) NO

DOMANDA 17

$f(n) = \Omega(g(n))$

$g(n) = \Omega(h(n))$

$\Omega(f(n)) \to 0 \le c\,g(n) \le f(n)$

COMBINARE LE DISUGUAGLIANZE

$\Omega(h(n)) \to 0 \le c\,h(n) \le g(n)$

$\Omega(g(n)) \to 0 \le c\,g(n) \le f(n)$

$[0 \le c_1 h(n) \le c_2 g(n) \le f(n)]$

$\to n \ge \max\{n_1, n_2\}$

$\{c_1 / c_2\} \to n$

$\forall n,\ f(n) = \Omega(h(n))$

$$\Theta(f(n)) = O(f(n)) \wedge \Omega(f(n))$$

DIMOSTRA SOPRA o SOTTO LIMITATI

$$\Theta = [0 \le c_1 f(n) \le g(n) \le c_2 f(n)]$$

$$O = [0 \le g(n) \le c_2 f(n)]$$

$$\Omega = [0 \le c_1 f(n) \le g(n)]$$

$$0 \le c_1 f(n) \le g(n) \le c_2 f(n)$$

$\rightarrow$ $n = \max\{n_1, n_2\}$

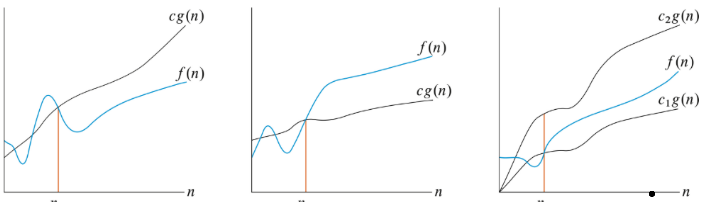
(VALE SEMPRE PER UN MAX)

$\forall n \ge n_0$

$n_0 \in \mathbb{N}$

The *asymptotic* efficiency of an algorithm is studied, that is, how its running time increases with the size of input *in the limit*, roughly speaking.

- $\Theta(g(n)) = \{f(n)$ : There exist positive constants $c_1$, $c_2$, and $n_0$ such that $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \ge n_0\}$. We say that g(n) is an *asymptotically tight bound* of f(n), and write $f(n) = \Theta(g(n))$. [It actually means $f(n) \in \Theta(g(n))$. The same for the following.]
- $O(g(n)) = \{f(n)$ : There exist positive constants c and $n_0$ such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0\}$. We say that g(n) is an *asymptotically upper bound* of f(n), and write $f(n) = O(g(n))$.
- $\Omega(g(n)) = \{f(n)$ : There exist positive constants c and $n_0$ such that $0 \le cg(n) \le f(n)$ for all $n \ge n_0\}$. We say that g(n) is an *asymptotically lower bound* of f(n), and write $f(n) = \Omega(g(n))$.
- $o(g(n)) = \{f(n)$ : For any positive constant $c > 0$, there exist a constant $n_0 > 0$ such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0\}$. We say that g(n) is an *upper bound* of f(n) but *not asymptotically tight*, and write $f(n) = o(g(n))$.
- $\omega(g(n)) = \{f(n)$ : For any positive constant $c > 0$, there exist a constant $n_0 > 0$ such that $0 \le cg(n) \le f(n)$ for all $n \ge n_0\}$. We say that g(n) is a *lower bound* of f(n) but *not asymptotically tight*, and write $f(n) = \omega(g(n))$.




$f(n) = O(g(n))$ (a)   $f(n) = \Omega(g(n))$ (b)   $f(n) = \Theta(g(n))$ (c)