Codifica di un messaggio

(NON AMBIGUO) Codice prefisso = codici in cui nessuna codifica è prefisso di un'altra

Abbiamo un alfabeto C{a,b,c,d}

Necessito di almeno 2 bit (2 alla 2 mi da 4) per codificare i caratteri del mio alfabeto:

es.
a = 00
b = 01
c = 10
d = 11

se ho un messaggio del tipo "a a a a"

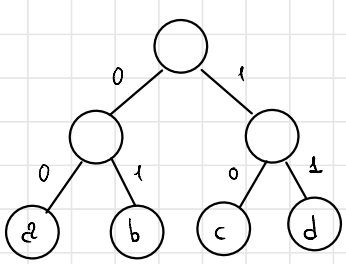
posso utilizzare una codifica del tipo :

a = 0
b = 01
c = 101
d = 11

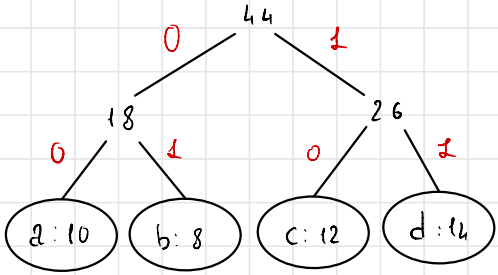
per risparmiare sul numero di bit usati però è una scelta valida solo per questo preciso messaggio ma non è una codifica valida per qualsiasi messaggio perchè creerebbe ambiguità nella decodifica dato che la dimensione per codificare i singoli caratteri non è fissa.

```
HUFFMAN (C, f)                    // O(n log n)
n ← |C|
Q ← MAKE-QUEUE (C, f)
FOR i ← 1 TO n-1 DO
    z ← NUOVO NODO INTERNO
    left (z) = x = EXTRACT-MIN(Q)
    right (z) = y = EXTRACT-MIN(Q)
    f(z) = f(x) + f(y)
    INSERT (Q, z, f(z))
RETURN EXTRACT-MIN(Q)
```

Si costruisce un albero di codifica cioè un albero binario dove le foglie rappresentano il messaggio che devo codificare



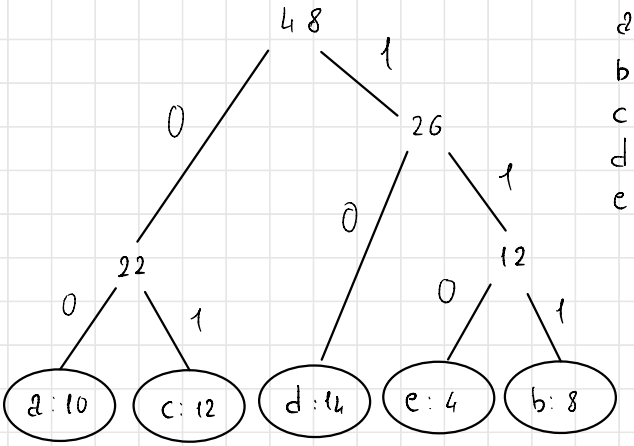
C1 = {a:10, b:8, c:12, d:14}



Però è dispendioso ricalcolare ogni volta i sottoproblemi (prog. dinamica) quindi penso di procedere calcolando ogni volta la somma degli elementi minori (scelta localmente ottima) al fine di calcolare la somma di tutti gli elementi e prendere quella di valore più basso.

C1 = {a:10, b:8, c:12, d:14 , e:4}

5 caratteri → 2^3 bit necessari



occorrenze bit

a: 00 → 10 · 2 = 20 ✓
b: 111 → 8 · 3 = 24 ✓
c: 01 → 12 · 2 = 24 ✓
d: 10 → 14 · 2 = 28 ✓
e: 110 → 4 · 3 = 12 =

108

↓ codifica variabile

mentre nella codifica fissa necessito di 3(bit) · 48 (somma delle occorrenze) = 144

Posso implementare l'estrazione dei due minimi con una min-Heap.

C++ dà la possibilità di implementare la tabella attraverso una tupla (in particolare una pair) per le associazioni carattere codifica

es. $\langle a, 00 \rangle$

a	00
b	111
c	01
d	10
e	110

pair <char, string>

Tabella di codifica T_{ab} { pair<...,...>, pair<...,...>, ..., pair<...,...> }

Huffman → H = T (in questo caso comprende solo T)

- Costruisce l'albero a partire dall'alfabeto
- Costruisce la tabella T di codifica
- Tenga la tabella e butta via tutto il resto

L'algoritmo di Huffman serve a costruire l'albero (albero di Huffman)