# Huffman

Problema della compressione dei dati

"Ridurre le dimensione del file"

Supponiamo di avere un testo T

$\Sigma$ :

| | |
|---|---|
| a : 20 | |
| b : 7 | |
| c : 3 | carattere : numero di occorrenze nel testo |
| d : 10 | |
| e : 15 | ogni carattere consuma 10 bit |
| f : 1 | |
| g : 30 | |
| h : 4 | |
| i : 5 | |
| l : 5 | |

$|T| = 100$

Per codificare tutto abbiamo bisogno: $100 \cdot 10 = 1000$ bit

"10 bit sono troppi"

| | | COD |
|---|---|---|
| a | 20 | 0000 |
| b | 7 | 0001 |
| c | 3 | 0010 |
| d | 10 | 0011 |
| e | 15 | 0100 |
| f | 1 | 0101 |
| g | 30 | 0110 |
| h | 4 | 0111 |
| i | 5 | 1000 |
| l | 5 | 1001 |

Codifichiamo "Dieci"

D i e c i

0011 - 1000 - 0100 - 0010 - 1000

a livello computazionale è istantanea questa codifica

Se la facilità di codifica e decodifica non è importante possiamo usare una dimensione dinamica per i nostri caratteri usando meno bit per i caratteri più frequenti e di più per quelli meno frequenti

| | COD | COD N |
|---|---|---|
| a : 20 | 0000 | |
| b : 7 | 0001 | |
| - c : 3 | 0010 | 01001 |
| - d : 10 | 0011 | 01 |
| - e : 15 | 0100 | 1 |
| f : 1 | 0101 | |
| g : 30 | 0110 | |
| h : 4 | 0111 | |
| - i : 5 | 1000 | 1100 |
| l : 5 | 1001 | |

La nuova codifica di "Dieci" è:

01110010100111000

17 bit

decifrare questa cosa è difficile perché non si sa la lunghezza fine

Dobbiamo stabilire dei codici fissi detti "prefissi" per rendere la decodifica fattibile, questi codici hanno queste caratteristiche:

- una codifica non deve avere un prefisso di un'altra codifica

| | COD | COD (seguendo ↗) |
|---|---|---|
| a : 20 | 0000 | 10 |
| b : 7 | 0001 | 1100 |
| c : 3 | 0010 | 1101 |
| d : 10 | 0011 | 000001 |
| e : 15 | 0100 | 001 |
| f : 1 | 0101 | 000000 |
| g : 30 | 0110 | 01 |
| h : 4 | 0111 | 111 |
| i : 5 | 1000 | 0001 |
| l : 5 | 1001 | 00001 |
| | 400 | 298 |

abbiamo perso un sacco di tempo per creare le nuove colonne

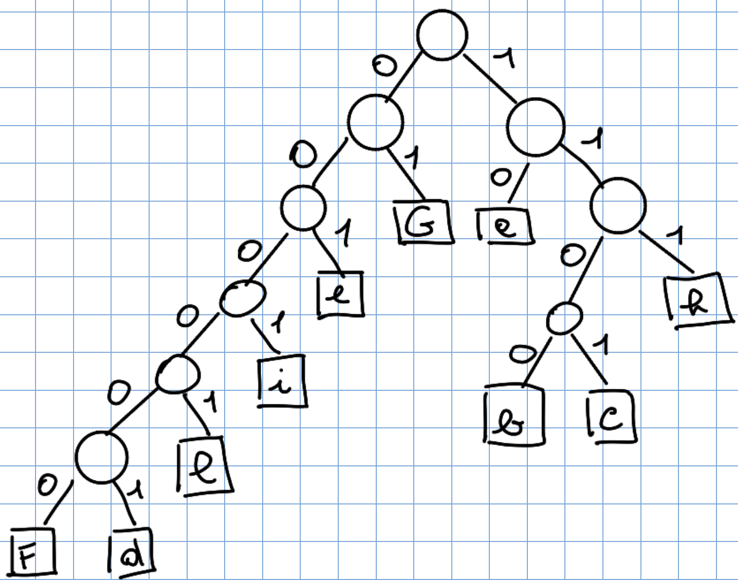
La codifica di "Dieci" diventa:

000001 0001 001 1101 0001

anche se ha dimensioni variabili riusciamo a decodificare

Huffman ci da un modo per creare la terza colonna in modo facile e la fa tramite un albero.

albero della nostra codifica:



L'obbiettivo diventa trovare l'albero che codifica meglio i nostri caratteri

$\forall c \in \Sigma$ (alfabeto) : $f(c) =$ freq

$d_T(c) =$ profondità

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

↓ Bonta di T

usiamo questa funzione per valutare il nostro albero

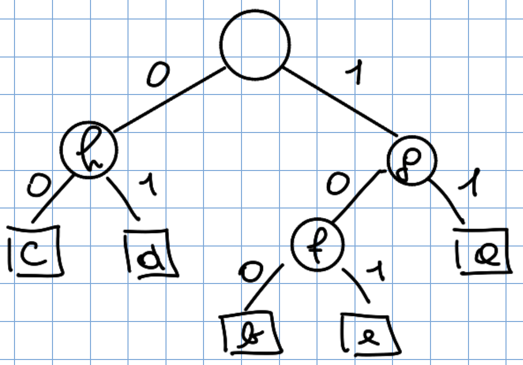
→ quanti bit usa l'albero T

Dobbiamo trovare una strategia che minimizza $B(T)$ lo faremo ricorsivamente, greedy.

Ma come riduciamo in sottoproblemi? Cerchiamo di definire un caso base:

Prendiamo due lettere del nostro alfabeto e le sostituisco con una

a
b
c
d
e
f
g
h




da questo albero ricaviamo i codici prefissi

Scegliamo 2 caratteri e li sostituiamo con 1 solo

Come troviamo i 2 caratteri migliori da scegliere?

Dimostrazione sottostruttura ottima $b(T) = b(T') + f(a) + f(b)$

$\Sigma$ soluzione $T$

$\Sigma' = \Sigma - \{a, b\} \cup \{z\}$ soluzione $T'$

$b(T) \sim b(T')$

$f(z) = f(a) + f(b)$

$d_T(a) = d_{T'}(z) + 1$

$d_T(b) = d_{T'}(z) + 1$

$$b(T) = \sum_{c \in \Sigma} f(c) \cdot d_T(c) = \left[\sum_{c \in \Sigma'} f(c)\, d_T(c)\right] - \underbrace{f(z)}_{f(a)+f(b)} \cdot d_{T'}(z) +$$

$$+ f(a)\, d_T(a) + f(b)\, d_T(b)$$

$d_T(z) + 1$

$= f(a) + f(b)$

$$-(\cancel{f(a)} + \cancel{f(b)})\, \cancel{d_{T'}(z)} + f(a)(\cancel{d_T(z)} + 1) - f(b)(\cancel{d_T(z)} + 1)$$

$$= f(a) + f(b)$$

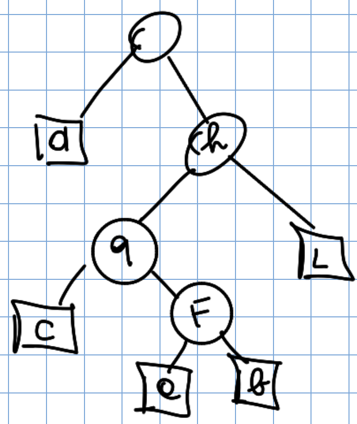
suppongo che la sottostruttura ottima non esiste

$\exists\, T'' \quad b(T'') < b(T')$

$b(T''') = b(T'') + f(a) + f(b)$

Come individuiamo le scelte migliore durante la costruzione dell'albero di Huffman

$a : 5$
$b : 5$
$c : 10$
$d : 40$
$e : 40$



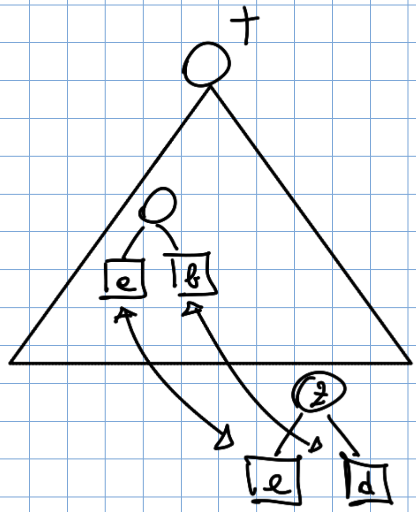

---

$\Sigma$

$a, b \in \Sigma$

$f(a) \leq f(c) \quad \forall c \in \Sigma$

$f(b) \leq f(c) \quad \forall c \in \Sigma - \{a\}$




$$B(T') = \sum_{c \in \Sigma} f(c)\, d_{T'}(c) = \sum_{c \in \Sigma - \{a, b, e, d\}}$$

$$f(c)\, d_{T'}(c) + \begin{cases} f(a)\, d_{T'}(a) \\ f(b)\, d_{T'}(b) \\ f(e)\, d_{T'}(a) \\ f(d)\, d_{T'}(b) \end{cases}$$

$$f(a)\, d_{T'}(a) + f(b)\, d_{T'}(b) + f(e)\, d_T(e) + f(d)\, d_{T'}(b)$$

$$d_{T'}(a)\big(F(a) + F(b)\big) + d_T(e)\big(F(e) + F(d)\big)$$

$B(T') \geq B(T)$ per ottenere una contraddizione

$B(T') - B(T) \leq 0$

$$B(T') - B(T) = f(a)\, d_{T'}(a) + f(b)\, d_{T'}(b) + f(e)\, d_{T'}(e) + f(d)\, d_{T'}(d)$$
$$= f(a)\, d_T(a) - f(b)\, d_T(b) - f(e)\, d_T\, e - F(d)\, d_T(d)$$

$f(e)\,(d_{t'}(e) - d_t(e))$

$f(e)\,(d_t(e) - d_{t'}(e)\,)$

$f(e) \cdot (d_{t'}(e) - d_t(e) - f(e)(d_{t'}(e) - d_t(e))$

$(d_{t'}(e) - d_t(e))(f(e) - f(e))$

```
HUFFMAN(Σ, f)
    Q ← new Priority Queue
    foreach c ∈ Σ do:              } n log n
        x ← new Node(c)
        insert x in Q
    for i ← 1 to |Σ|-1 Do                          } n log n
        x ← extractMin(Q)          } log n
        y ← extractMin(Q)
        z ← new Node
        f(z) = f(x) + f(y)
        left(z) = x
        right(z) = y
        insert z in Q
```

$n = |\Sigma|$

~~m: 4~~

~~n: 8~~

~~o: 10~~

~~p: 15~~

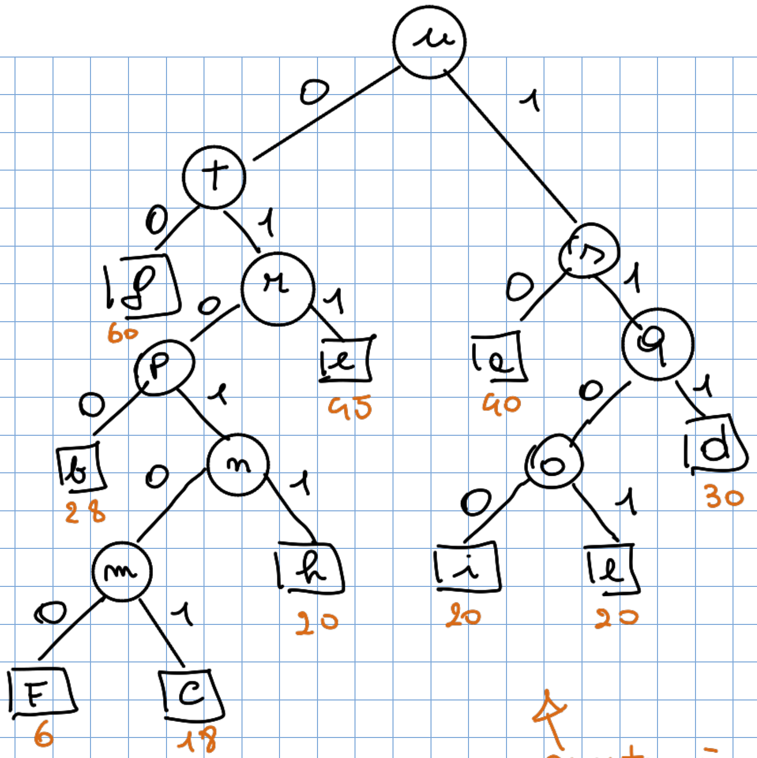
~~q: 20~~

~~r: 30~~

~~s: 40~~

~~t: 60~~

u: 100




questo è l'albero ottimo

| | COD | COD |
|---|---|---|
| a : 20 | 0000 | 10 |
| b : 7 | 0001 | 1100 |
| c : 3 | 0010 | 1101 |
| d : 10 | 0011 | 000001 |
| e : 15 | 0100 | 001 |
| f : 1 | 0101 | 000000 |
| g : 30 | 0110 | 01 |
| h : 4 | 0111 | 111 |
| i : 5 | 1000 | 0001 |
| l : 5 | 1001 | 00001 |
| | 400 | 298 |