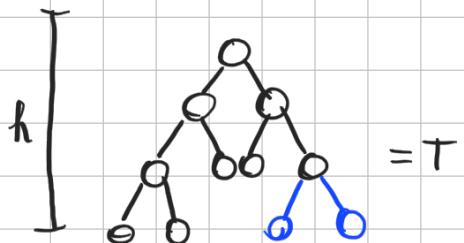


## PROPOSIZIONE

l'albero che minimizza l'altezza minimizza anche il percorso esterno (in alberi completi)

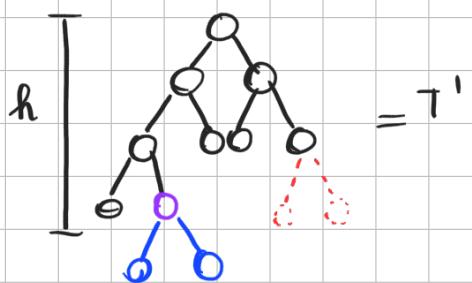
## DI MOSTRAZIONE

Prendo un albero



l'unica cosa che potremmo fare è "spostare i nodi".

Vediamo cosa accade



$T'$  mantiene tutte le proprietà.

Se  $PE(T) = x$  (percorso esterno) allora:

$$PE(T') = x - 2h + (h-1) - h + 2(h+1) = x+1$$

Se ogni modifica di  $T$  peggiora  $PE$ , allora  $T$  minimizza  $PE$

## RICERCA MIGLIOR CASO MEDIO

Ricordiamo che per un albero completo (che abbiamo visto minimizza il percorso esterno e l'altezza) le foglie sono a profondità  $h = h-1$ . E' facile calcolare PE che sarà:

$$PE(T) = N_h \cdot h + N_{h-1} \cdot (h-1)$$

dove  $N$  è il numero di foglie.

Troviamo il valore dell'espressione sappiendo che:

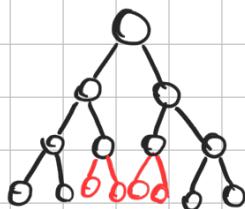
$$\textcircled{1} \quad N_h + N_{h-1} = m! \quad (\text{numero di foglie totali})$$

$$\textcircled{2} \quad \text{In un albero pieno sappiamo che il numero di foglie è } 2^h.$$

l'albero pieno contiene l'albero completo, quindi possiamo dedurre

$$\text{che } 2^h = N_h + 2N_{h-1} \quad (2N_{h-1} \text{ perché ad ogni foglia a livello}$$

$h-1$  sostituiamo un nodo interno che abbia due figli)



Quindi abbiamo ora un sistema di due equazioni in due incognite

$$\begin{cases} N_h + N_{h-1} = m! \\ 2N_{h-1} + N_h = 2^h \end{cases} \rightarrow N_h + 2(m! - N_h) = 2^h \rightarrow N_h + 2m! - 2N_h = 2^h$$

$$\rightarrow N_h = 2m! - 2^h$$

$$\rightarrow N_{h-1} = m! - (2m! - 2^h) = 2^h - m!$$

Calcoliamo il percorso esterno:

$$PE = h \cdot (2m! - 2^h) + (h-1)(2^h - m!)$$

Vogliamo il tempo medio che era dato da  $\frac{PE}{m!}$  quindi

$$\frac{PE}{m!} = \frac{h \cdot (2m! - 2^h)}{m!} + \frac{(h-1)(2^h - m!)}{m!} = \frac{2hm! - h2^h + h2^h - hm! - 2^h + m!}{m!}$$
$$= \frac{hm! - 2^h + m!}{m!} = h - \frac{2^h}{m!} + 1$$

Non conosciamo l'altezza in funzione dei modi, in questo caso non abbiamo il numero di modi bensì quello delle foglie, possiamo comunque ricavene un'upperbound:

Abbiamo precedentemente dimostrato che  $h = \lceil \lg_2 m! \rceil$  (si può dimostrare che  $h = \lfloor \lg_2 m! \rfloor + \lceil \lg_2 m! \rceil$  per un albero pieno) quindi

$$\frac{PE}{m!} = h - \frac{\lceil \lg_2 m! \rceil}{m!} + 1 = h + \Theta(1) + 1$$

Ma abbiamo visto che  $h = \Omega(m \lg m)$ , quindi l'albero migliore ha almeno tempo  $m \lg m$

In definitiva, VT di ordine  $m$

$$T_H(T) = \Omega(m \lg m)$$

## DI MOSTRAZIONE

Dimostriamo con un secondo metodo che  $\lg m! = \Theta(m \lg m)$

$$\lg m! = \lg \left( \prod_{i=1}^m i \right) = \sum_{i=1}^m \lg i$$

$$\sum_{i=1}^m \lg i = \Theta(m \lg m) \text{ perche'}$$

$$m \lg m = \frac{m}{2} \lg \left( \frac{m}{2} \right) = \sum_{i=2}^m \lg \left( \frac{m}{2} \right) \leq \sum_{i=2}^m \lg i \leq \sum_{i=1}^m \lg i \leq \sum_{i=1}^m \lg m = O(m \lg m)$$

## RAPPRESENTAZIONE DEGLI INSIEMI

Per rappresentare un insieme  $S$  useremo un **ARRAY** tale che  
 $\text{length}(A) \geq |S|$  ( $A$  potrebbe avere duplicati)

## RICERCA SU STRUTTURE DATI

In termini di funzioni, l'algoritmo di ricerca di un elemento  $a$  in un insieme  $S$  è data dalla **FUNZIONE CARATTERISTICA**

$$(S, k) = \begin{cases} 1 & k \in S \\ 0 & k \notin S \end{cases}$$

Vediamo quanto costa la ricerca in un array in funzione di  $|S|$  ricordando che se  $A$  è l'array  $|A| \geq |S|$  come detto prima.

Nel caso peggiore  $T_{\text{SEARCH}} = O(|S|)$  borbulente ( $k \in S$ ).

## OSS

Da adesso in poi assumeremo che non ci siano doppiioni nell'array e che  $|S| = m$

## INSEGNAMENTO IN UN ARRAY

Vediamo l'array come una coppia  $(A, i)$  dove  $A$  è l'array e  $i$  è l'indice della prima cella libera dell'array.

Quando inseriamo un elemento avremo una nuova struttura  $(A', i')$  dove  $A' = A \cup \{k\}$  ( $k$  è l'elemento aggiunto) e  $i' = i + 1$ .

Abbiamo un problema quando l'array si riempie, implicando la creazione di un nuovo array con i primi  $i$  elementi copiati e la dimensione dell'array aumentata, implicando un costo considerevole.