Algoritmi e Strutture Dati

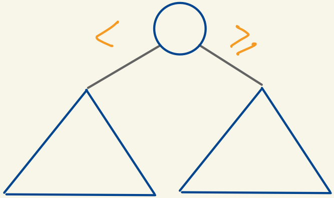
Lezione 27

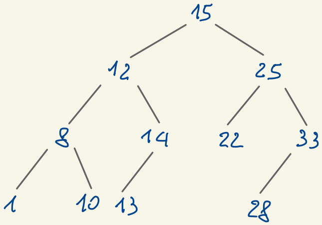
24 novembre 2025

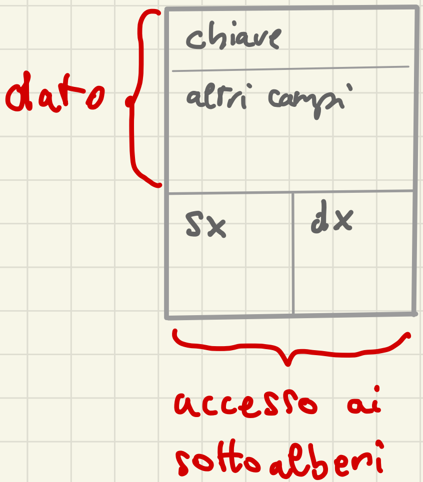
Alberi binari di ricerca

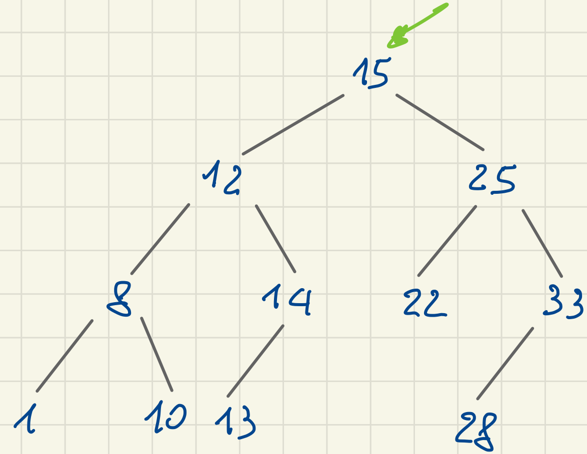
Un *albero binario di ricerca* è un albero binario in cui *per ogni nodo n*:

- il valore di ogni chiave contenuta nel sottoalbero sinistro di *n* è minore della chiave contenuta in *n*,
- il valore di ogni chiave contenuta nel sottoalbero destro di *n* è maggiore o uguale della chiave contenuta in *n*.

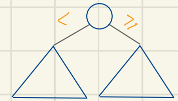





dato
chiave
altri campi
sx
dx
accesso ai sottoalberi

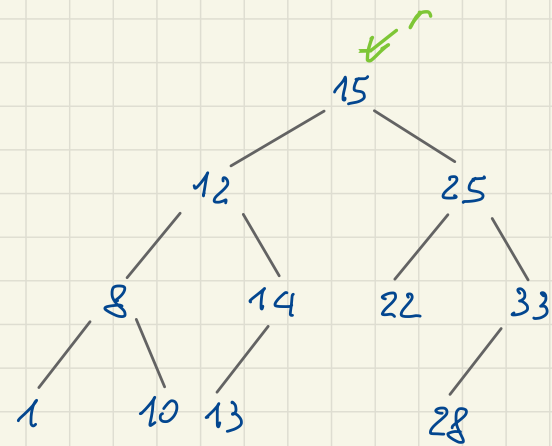

15
12
25
8
14
22
33
1
10
13
28

Ricerca ricorsiva

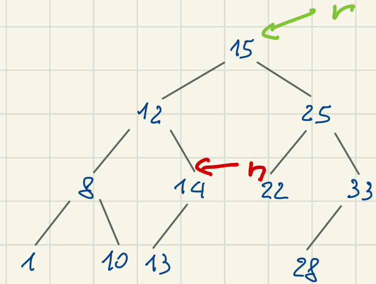



```
FUNZIONE trova (AlberoRicerca r, TipoChiave k) → Nodo
    IF r = null THEN
        RETURN null
    ELSE IF k < r.chiave THEN
        RETURN trova (r.sx, k)
    ELSE IF k > r.chiave THEN
        RETURN trova (r.dx, k)
    ELSE RETURN r
```

ricorsioni in coda



Ricerca iterativa



k 14

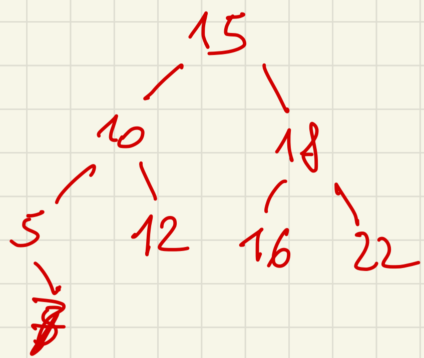
```
FUNZIONE trova(AlberoRicerca r,
        tipoChiave k) → Nodo

   n ← r
   WHILE n ≠ null AND n.chiave ≠ k DO
      IF k < n.chiave THEN n ← n.sx
                      ELSE n ← n.dx
   RETURN n
```

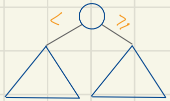
Inserimento

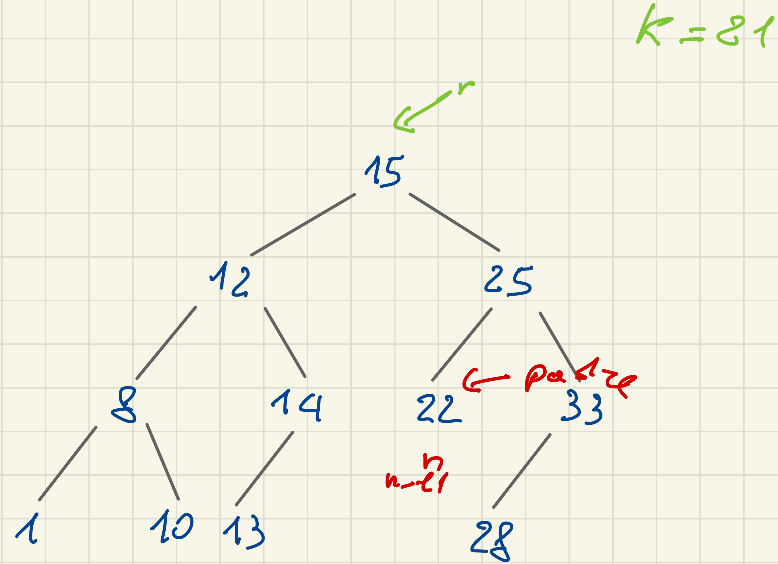
Disegnare l'albero di ricerca che si ottiene a partire da un albero vuoto inserendo uno dopo l'altro, nell'ordine indicato, i seguenti numeri:

15 10 5 18 12 7 16 22

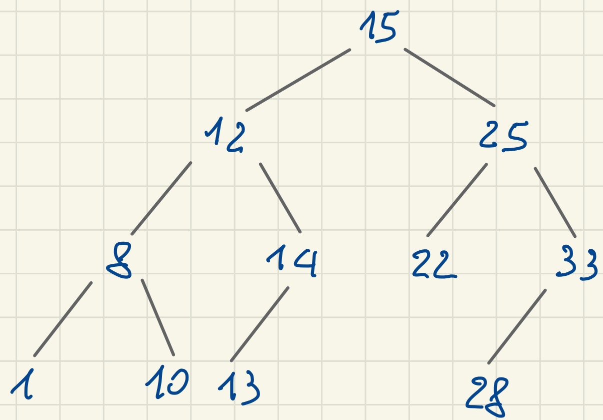


Inserimento ricorsivo




k = 21
r
15
12
25
8
14
22
33
1
10
13
28

Inserimento iterativo



FUNZIONE inserisci (AlberoRicerca r, elemento d) → AlberoRicerca

```
k ← d.chiave
t ← rif. a un nuovo nodo
t.chiave ← k
t.altri campi ← d.altri campi
t.sx ← null
t.dx ← null
```

creazione nuovo nodo

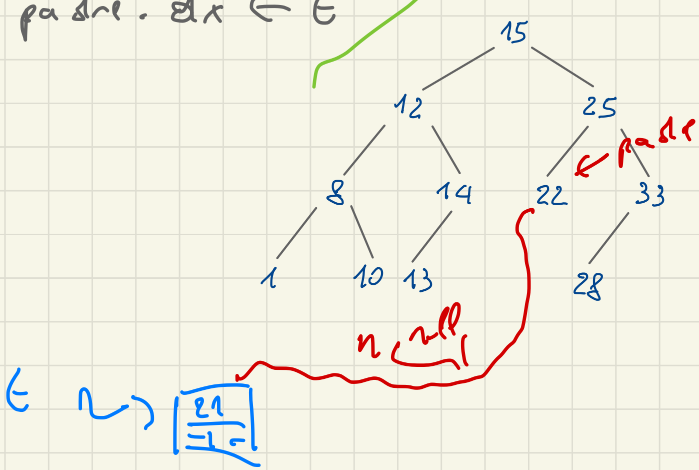
```
padre ← null
n ← r
WHILE n ≠ null AND n.chiave ≠ k DO
    padre ← n
    IF k < n.chiave THEN n ← n.sx
                    ELSE n ← n.dx
```

ricerca posizione

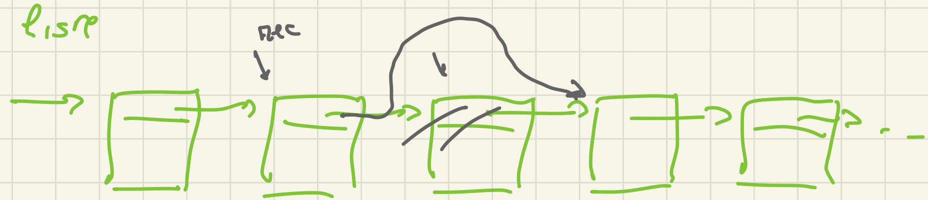
```
IF padre = null THEN
    r ← t
ELSE IF k < padre.chiave THEN
        padre.sx ← t
    ELSE padre.dx ← t
```

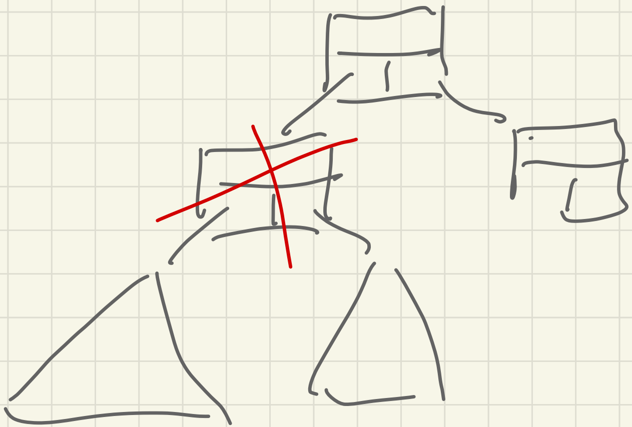
inserimento

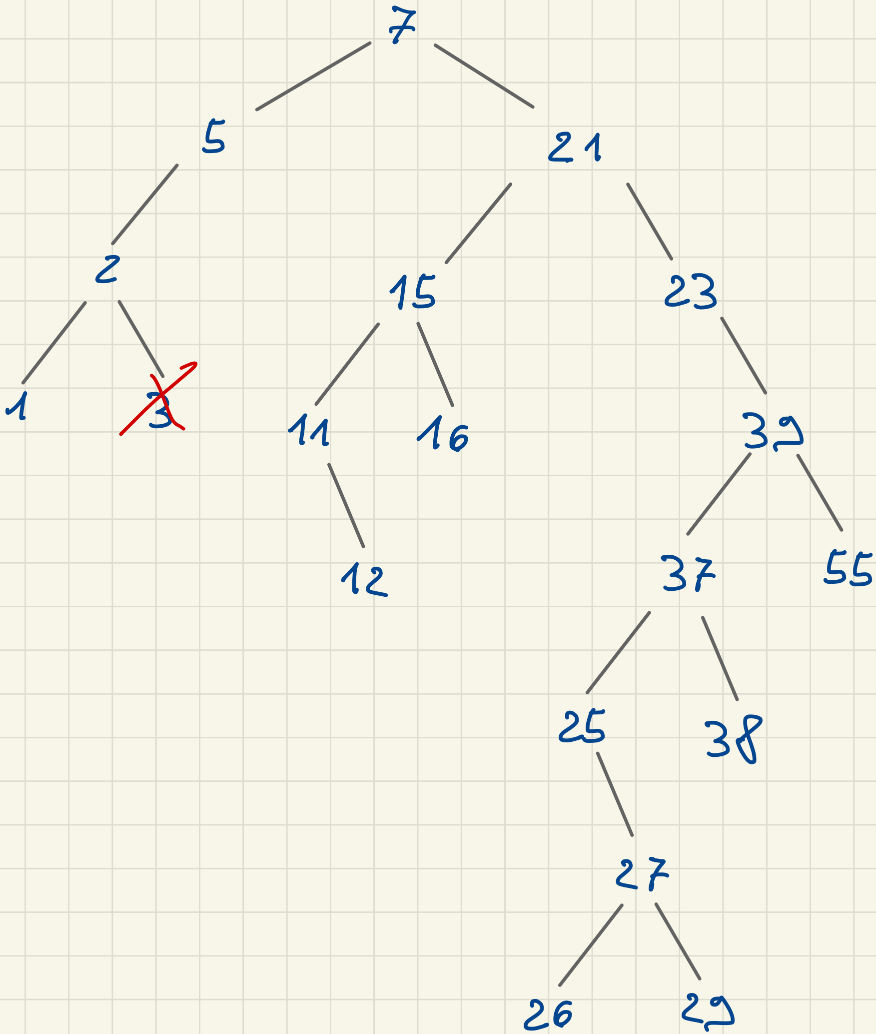
```
RETURN r
```



Cancellazione







Cancella 3

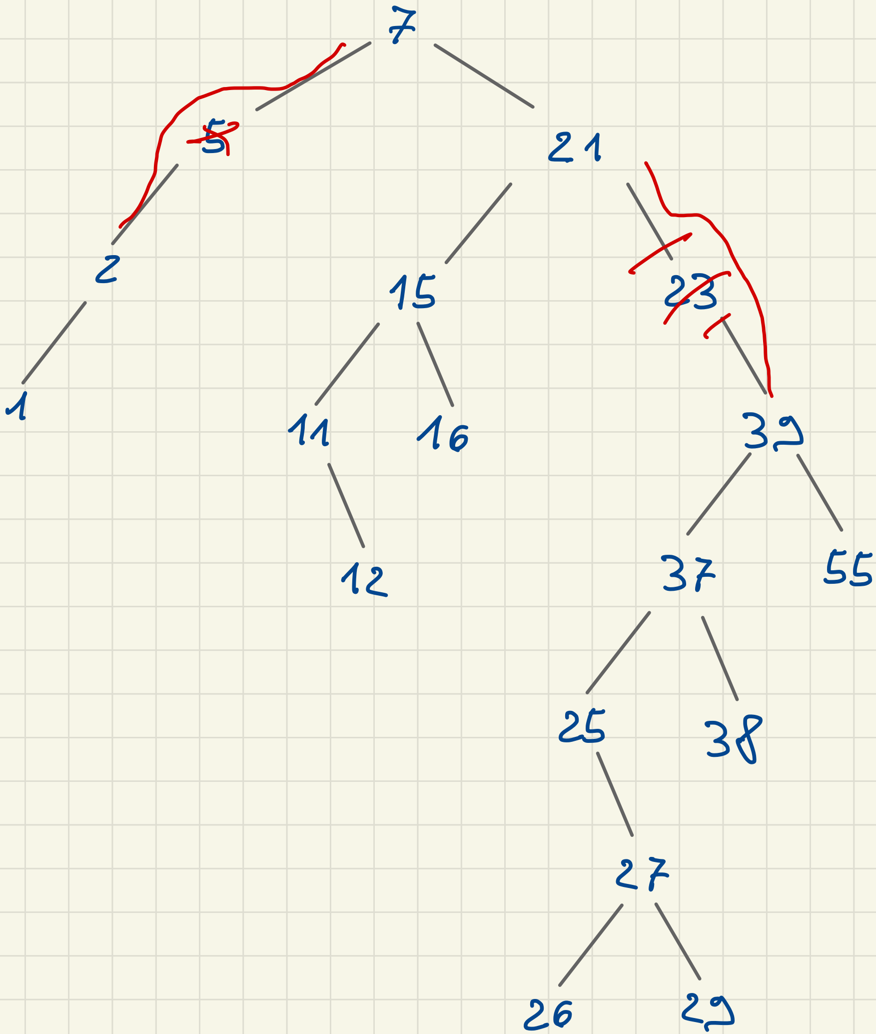
Caso 1

il nodo è una foglia

FACILE

caso particolare:

cancellazione radice



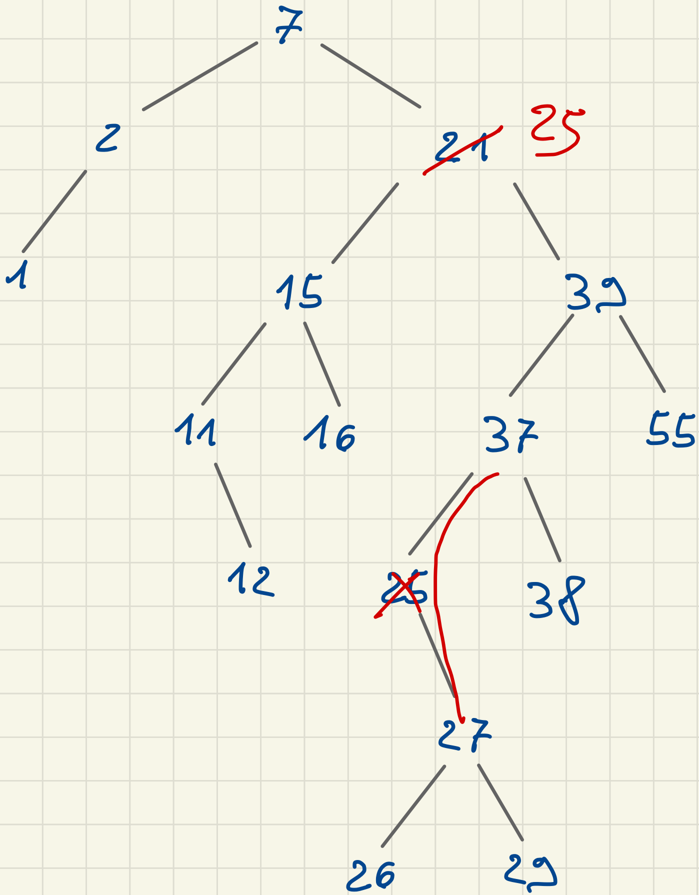
Cancella 23

Cancella 5

Caso 2 il nodo da cancellare ha un unico figlio:

Modifica il puntatore nel nodo padre

CASO PARTICOLARE: cancellare radice

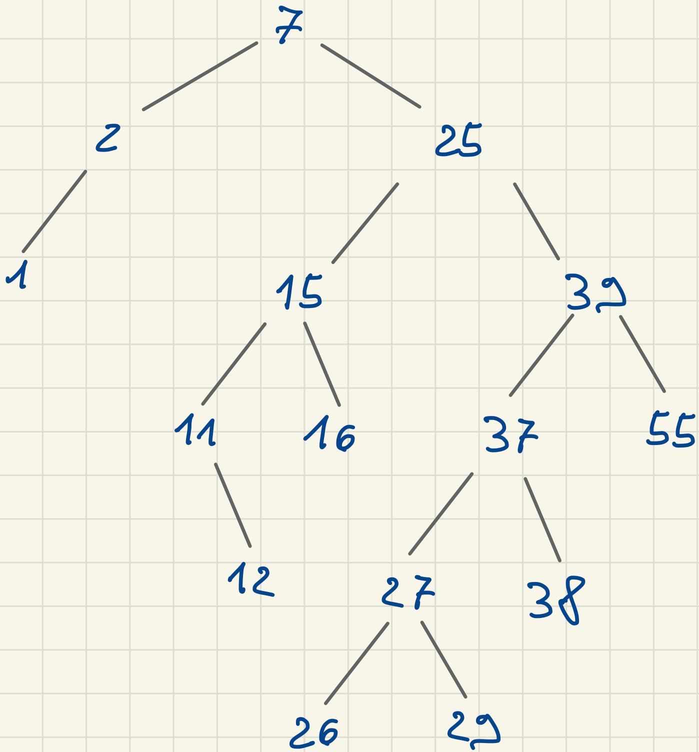


Cancella 21

sostituisci con il contenuto del nodo con chiave più piccola del sottoalbero dx del nodo da cancellare

il nodo eliminato ha al max 1 figlio

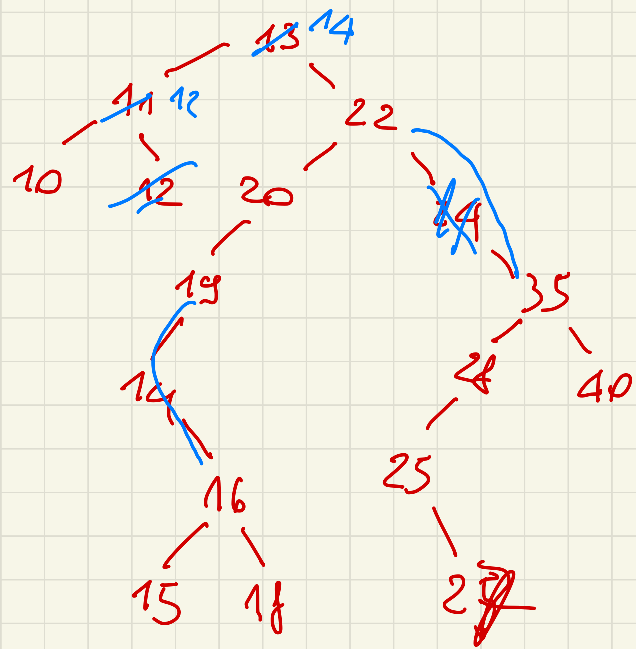
↪ rimuovilo (caso 1 o 2)


7
2
25
1
15
39
11
16
37
55
12
27
38
26
29

Esercizio

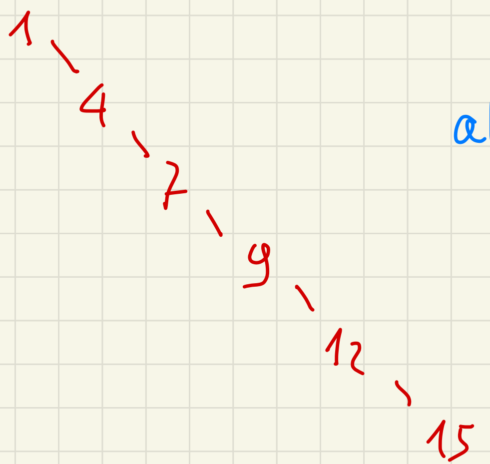
Disegnare l'albero di ricerca che si ottiene a partire da un albero vuoto inserendo uno dopo l'altro, nell'ordine indicato, i seguenti numeri: 13 22 20 11 24 35 28 40 19 14 10 12 16 25 15 27 18

Disegnare come diventa l'albero se si rimuovono 13 24 11



Costo inserimento
ricerca
cancellazione $\Theta(h)$

1 4 7 9 12 15

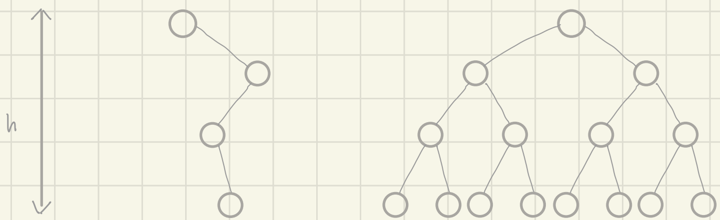


albero degenere $h \approx n$

Alberi binari: $\underbrace{\text{n° nodi}}_{n}$ vs $\underbrace{\text{altezza}}_{h}$

Che relazioni ci sono tra numero di nodi e altezza in un albero binario?

Alberi binari: $\underbrace{n^\circ \text{ nodi}}_{n}$ vs $\underbrace{\text{altezza}}_{h}$



$h+1$ albero degenere

$2^{h+1}-1$ albero completo

$$h+1 \le n \le 2^{h+1}-1$$

$h \le n-1$

$h < n$

$n < 2^{h+1}$

$\log_2 n < h+1$

$\lfloor \log_2 n \rfloor \le h$

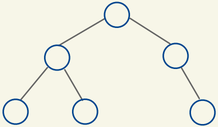
$$\lfloor \log_2 n \rfloor \le h < n$$

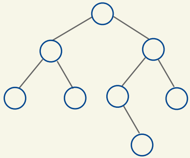
Alberi bilanciati

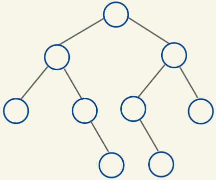
Alberi perfettamente bilanciati

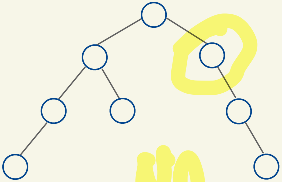
Definizione

Un albero binario è detto *perfettamente bilanciato* quando *per ogni nodo* la differenza in valore assoluto tra i numeri di nodi presenti nei suoi sottoalberi sinistro e destro è al massimo 1

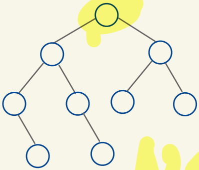








NO



NO

Alberi perfettamente bilanciati: $\underbrace{\text{n° nodi}}_{n}$ vs $\underbrace{\text{altezza}}_{h}$ $\Rightarrow$ ALTEZZA SEMPRE LOGARITMICA!

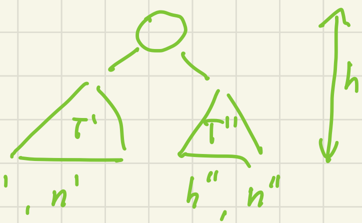
$\boxed{A}$ n° minimo di nodi

$$n \geq 2^h$$

Dim induzione su h

Base $\boxed{h=0}$ • 1 nodo $1 \geq 2^0$ ok

Induzione $\boxed{h-1 \to h}$



- almeno uno tra T' e T'' deve avere altezza $h-1$
- supponiamo sia T'

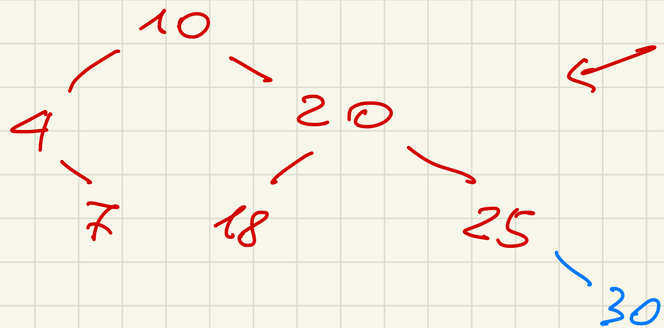
per ip. induzione $n' \geq 2^{h'} = 2^{h-1}$

poiché T è perf. bilanciato: $n'' \geq n' - 1$

$$n = 1 + n' + n'' \geq \cancel{1} + 2^{h-1} + 2^{h-1} - \cancel{1} = 2 \cdot 2^{h-1} = 2^h \qquad \square$$

$\boxed{B}$ $n < 2^{h+1}$ (n° max nodi: albero completo)

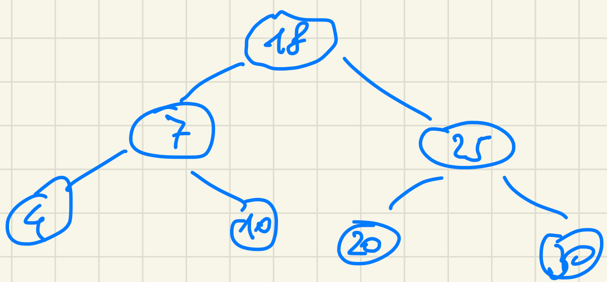
$$2^h \leq n < 2^{h+1} \qquad h \leq \log_2 n < h+1 \Rightarrow h = \lfloor \log_2 n \rfloor$$



perf. s.l.

non perf. s.l.

4 7 10 18 20 25 30



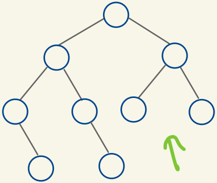
l'inserimento può chiedere la ristrutturazione dell'intero albero

$\Rightarrow$ TROPPO COSTOSO

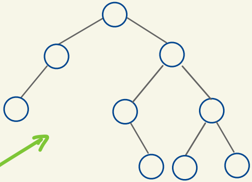
Alberi bilanciati in altezza

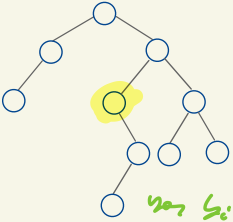
Definizione

Un albero binario è detto *bilanciato* (in altezza) o AVL[1] quando *per ogni nodo* la differenza in valore assoluto tra le altezze dei suoi sottoalberi sinistro e destro è al massimo 1

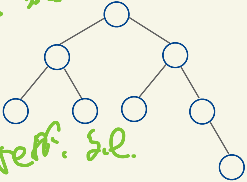


bilanciato
ma non perf. bil

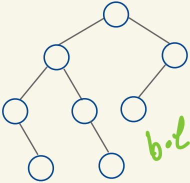




non bil



perf. bil.



bilanc.

NOTA perf. bil ⇒ bilanc.

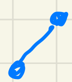
[1] Adelson-Velsky and Landis, 1962

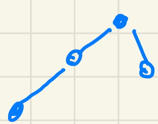
Alberi bilanciati in altezza: $\underbrace{\text{n° nodi}}_{n}$ vs $\underbrace{\text{altezza}}_{h}$

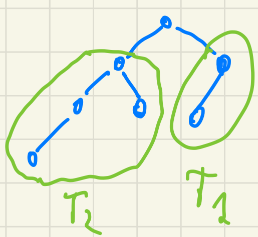
n° max di nodi → albero completo di altezza h : $n = 2^{h+1} - 1$

n° min di nodi

$h = 0$  T_0 $n_0 = 1$

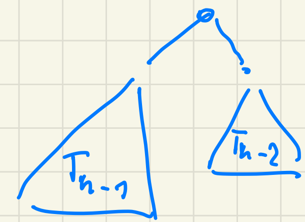
$h = 1$  T_1 $n_1 = 2$

$h = 2$  T_2 $n_2 = 4$

$h = 3$ 

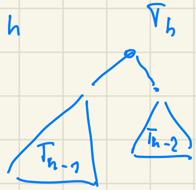
T_3 $n_3 = 7$

h T_h



ALBERO di FIBONACCI di ALTEZZA h

Albero AVL di altezza minima



ALBERO di FIBONACCI di ALTEZZA h

$n_h =$ # nodi di T_h

$$n_h = \begin{cases} 1 & \text{se } h=0 \\ 2 & \text{se } h=1 \\ 1 + n_{h-1} + n_{h-2} & \text{altrimenti} \end{cases}$$

Dim che

$$n_h = F_{h+3} - 1$$

$F_1 = F_2 = 1$

$F_k = F_{k-1} + F_{k-2}$

Ind. $h = 1$ $n_0 = 1$ $F_3 = 2$

$h = 2$ $n_1 = 2$ $F_4 = 3$ | ok base

$< h \to h$

$$n_h = 1 + n_{h-1} + n_{h-2}$$

ip. ind $\to$ $= \underbrace{\cancel{1} + F_{h+2} \cancel{-1} + F_{h+1}} - 1$

$$= F_{h+3} - 1 \quad \square$$

$$n_h \approx \frac{\phi^{h+3}}{\sqrt{5}} \qquad \sqrt{5}\, n_h \approx \phi^{h+3}$$

$$\log_\phi(\sqrt{5}\, n_h) \approx h+3 \qquad h \approx \log_\phi \sqrt{5} - 3 + \log_\phi(n_h)$$

$h = \Theta(\log n) \to$ Ricerca $\Theta(\log n)$