

Algoritmi e Strutture Dati

Lezione 29

28 novembre 2025

B-Alberi

Dati su memoria secondaria

- DBMS** Data Base Management System:
uso di indici per garantire accesso veloce ai dati
- Possibili più indici, basati su campi differenti
- Indici** Possono essere molto grandi (migliaia o milioni di chiavi).
- A causa delle loro dimensioni non possono essere caricati in memoria centrale
- Dizionari** Si possono realizzare strutture ad albero per implementare dizionari (per rappresentare indici) in *memoria secondaria*
- È necessario tenere conto di *vincoli tecnologici*

Memoria secondaria: vincoli tecnologici



- Accesso ai dati *più lento* che in memoria centrale

Disco: tempi di latenza dovuti a parti meccaniche,
tempo di accesso circa 100.000 volte più grande rispetto all'accesso
in memoria centrale

Accesso al singolo dato più lento

- Accesso a *blocchi* (o pagine)

Dimensioni tipiche: $2^9 = 512$ byte, ... $2^{12} = 4096$ byte

In un unico accesso si ottengono molti dati

B - alberi

alberi & Bayer

B-Alberi

6

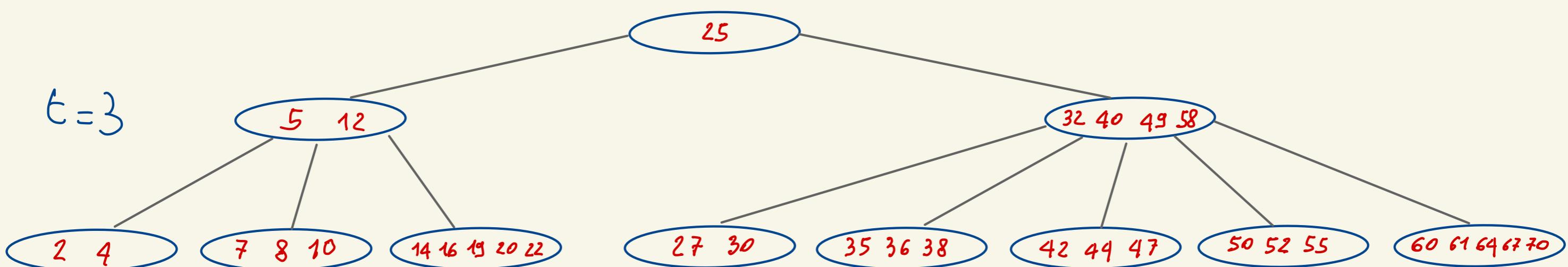
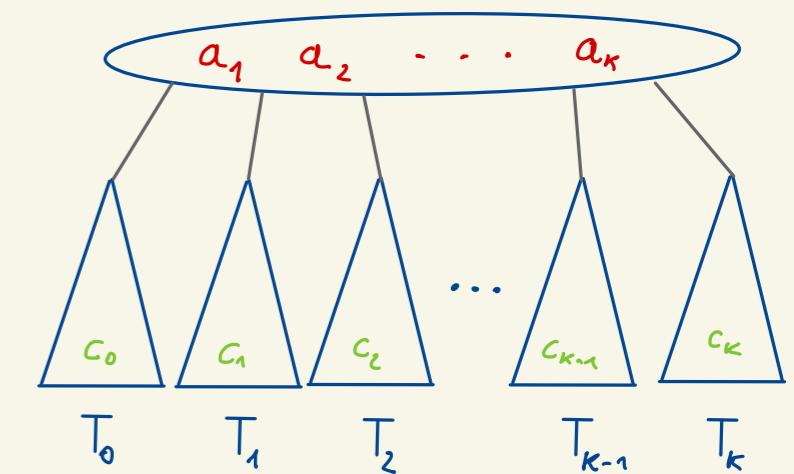
Definizione

Un **B-albero di ordine t** (grado minimo t) è un albero T tale che:

- ogni *nodo interno* ha al massimo $2t$ figli,
 - ogni *nodo interno diverso dalla radice* ha almeno t figli,
 - la radice ha almeno 2 figli,
 - tutte le foglie si trovano allo stesso livello,
- oppure
- T è l'albero vuoto.

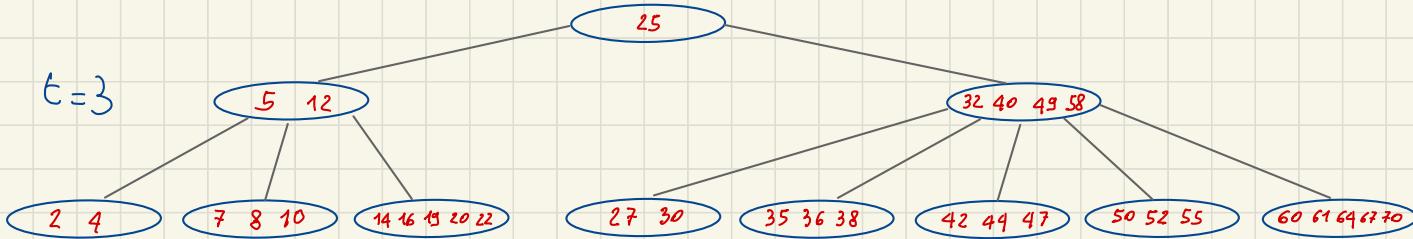
- Ogni *foglia* contiene k chiavi ordinate $a_1 < a_2 < \dots < a_k$ dove $t - 1 \leq k \leq 2t - 1$
($1 \leq k \leq 2t - 1$ se la foglia è la radice)
- Ogni *nodo interno* con $k + 1$ figli e sottoalberi T_0, T_1, \dots, T_k contiene k chiavi ordinate $a_1 < a_2 < \dots < a_k$ tali che, per ogni chiave c_i di T_i , $i = 0, \dots, k$, risulta

$$c_0 < a_1 < c_1 < a_2 < \dots < c_{k-1} < a_k < c_k$$



Ricerca

8 30 53



- **Nodo :** array ordinato con k chiavi $k < 2t$



Ricerca binaria

$\Theta(\log t)$ punti

chiave aspettata

chiave presente → finito

nodo interno

Foglio → finito

si posseguono
su sotto-albero

- Altezza dell'albero h → si ripete al massimo h volte

Torcate di passo

$\Theta(h \log t)$

Quanto può essere al massimo h se ci sono n chiavi?

$$h=0$$

$$h=1$$

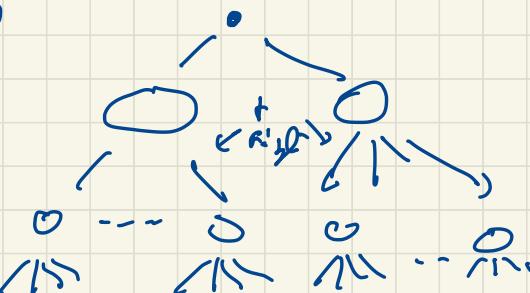
$$h=2$$

$$h=3$$

$$\vdots$$

$$= 1 + 2(t-1) \sum_{i=0}^{h-1} t^i$$

$$\Rightarrow n \geq 2t^h - 1$$



nodhi

$$1$$

$$+ 2$$

$$+ 2t$$

$$+ 2t^2$$

:

$$+ 2t^{h-1}$$

n chiavi

$$1$$

$$+ 2(t-1)$$

$$+ 2t(t-2)$$

$$+ 2t^2(t-1)$$

$$+ 2t^{h-1}(t-1)$$

n chiavi
minimo per albero h

$$\frac{1 + 2 \sum_{i=0}^{h-1} t^i (t-1)}{t-2} =$$

$$\frac{t^h - 1}{t-2}$$

$$= 1 + 2t^{h-2} \approx 2t^{h-1}$$

$$n \geq 2t^h - 1$$
$$t^h \leq \frac{n+1}{2}$$
$$h \leq \log_2 \frac{n+1}{2}$$

Se ci sono n chiavi l'altezza h è

il massimo

$$\log_2 \frac{n+1}{2}$$

passi totali per ricerca $\Theta(h \lg t)$

$$h \leq \lg_t \frac{n+1}{2}$$

$$h \lg_b t \leq \lg_t \frac{n+1}{2} \cdot \lg_b t$$

$$= \lg_b \frac{n+1}{2}$$

passi di ricerca $\Theta(\lg n)$

indipendente da t

$$\left\{ \begin{array}{l} \lg_b x = \frac{\lg_a x}{\lg_a b} = \\ = \lg_a x \cdot \lg_b a \end{array} \right.$$

Accessi a memoria secondaria in B-alberi

- Scegliamo t in modo che ogni nodo sia contenuto INTERAMENTE in un blocco

- n° di accessi \approx altezza albero
 $\approx \lg_t n$

Se usassi alberi AVL o 2-3? n° accessi $\propto \lg_2 n$

$$10^6 = 1 \text{ milione di chiam}$$

$$\approx 2^{20}$$

$$\text{AVL } \lg_2 10^6 \approx \lg_2 2^{20} \approx 20$$

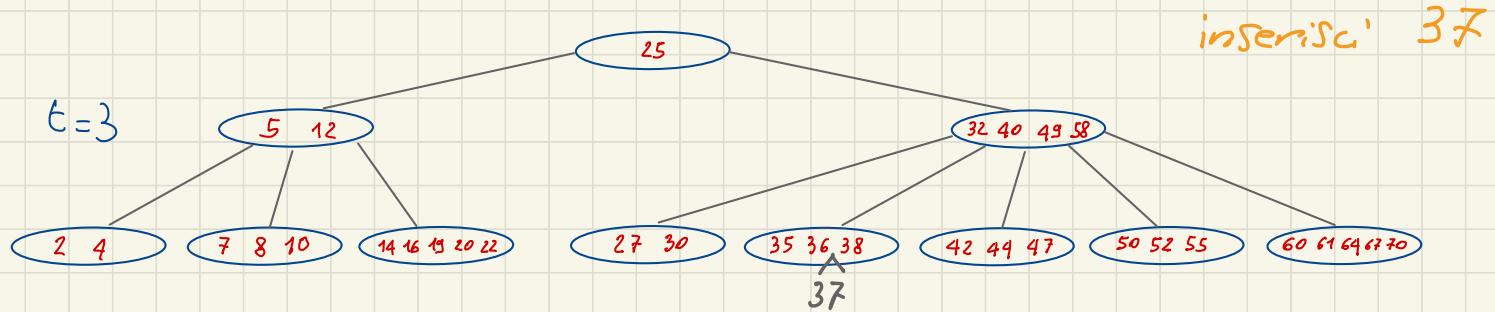
accessi
a massimo

$$\text{B-albero } t = 100$$

$$\# \text{accessi} \approx \lg_{100} 10^6 = 3$$

Inserimento

- Ricerca la foglia in cui l'elemento va inserito



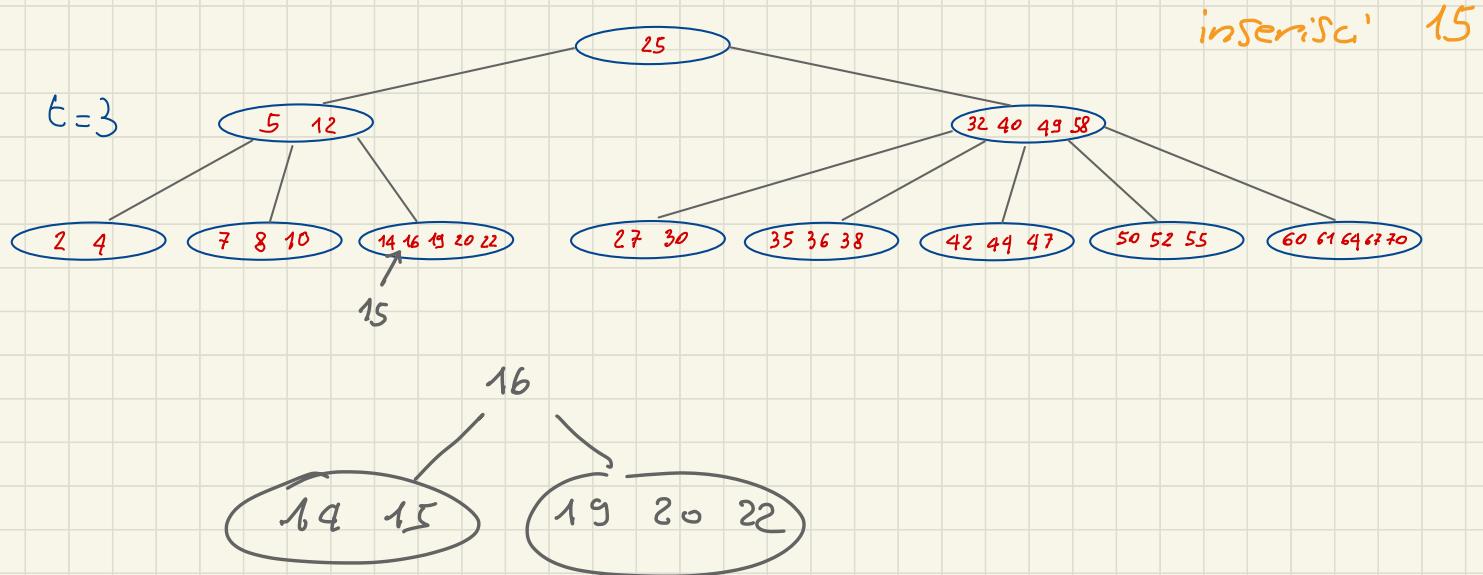
Caso 1 C'è spazio nell foglia



inserisci l'elemento al posto giusto

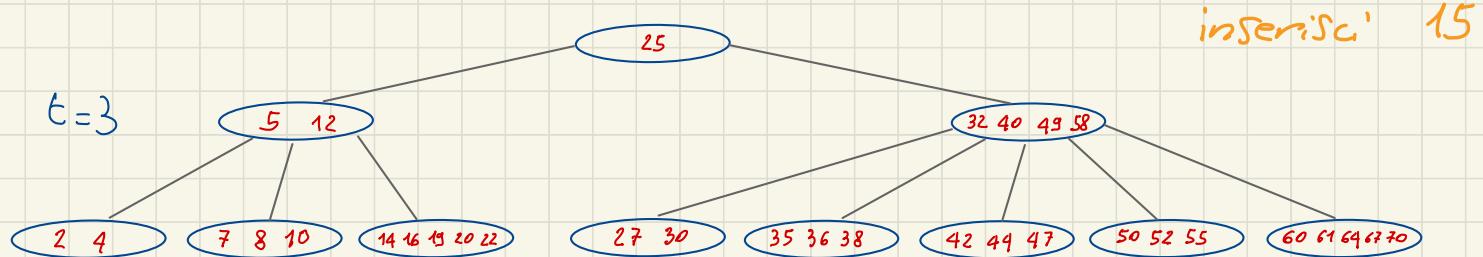
Inserimento

- Ricerca la foglia in cui l'elemento va inserito

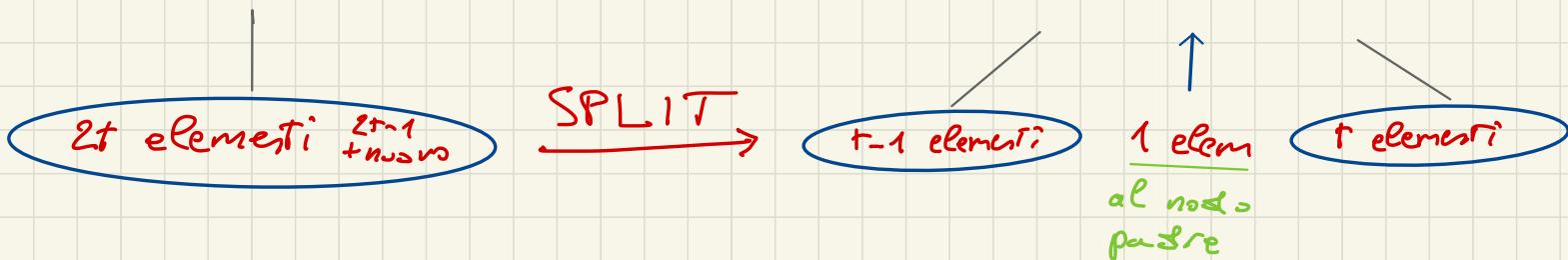


Inserimento

- Ricerca la foglia in cui l'elemento va inserito

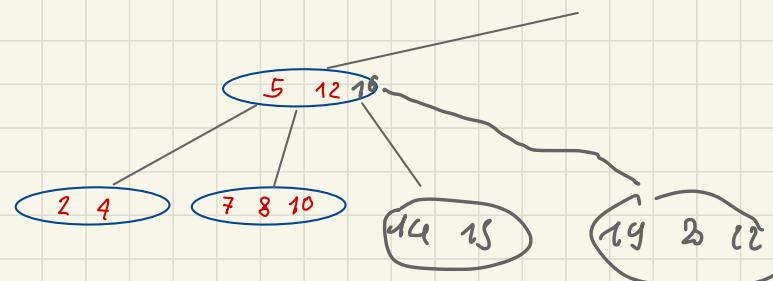
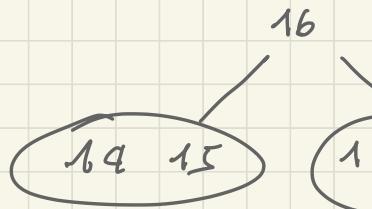
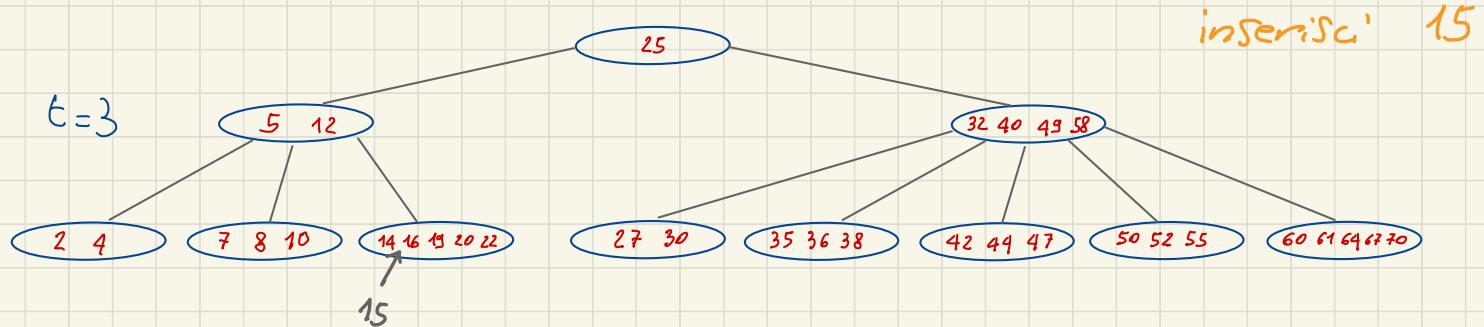


Caso 2 Non c'è spazio nella foglia (ci sono già $2t-1$ elementi)

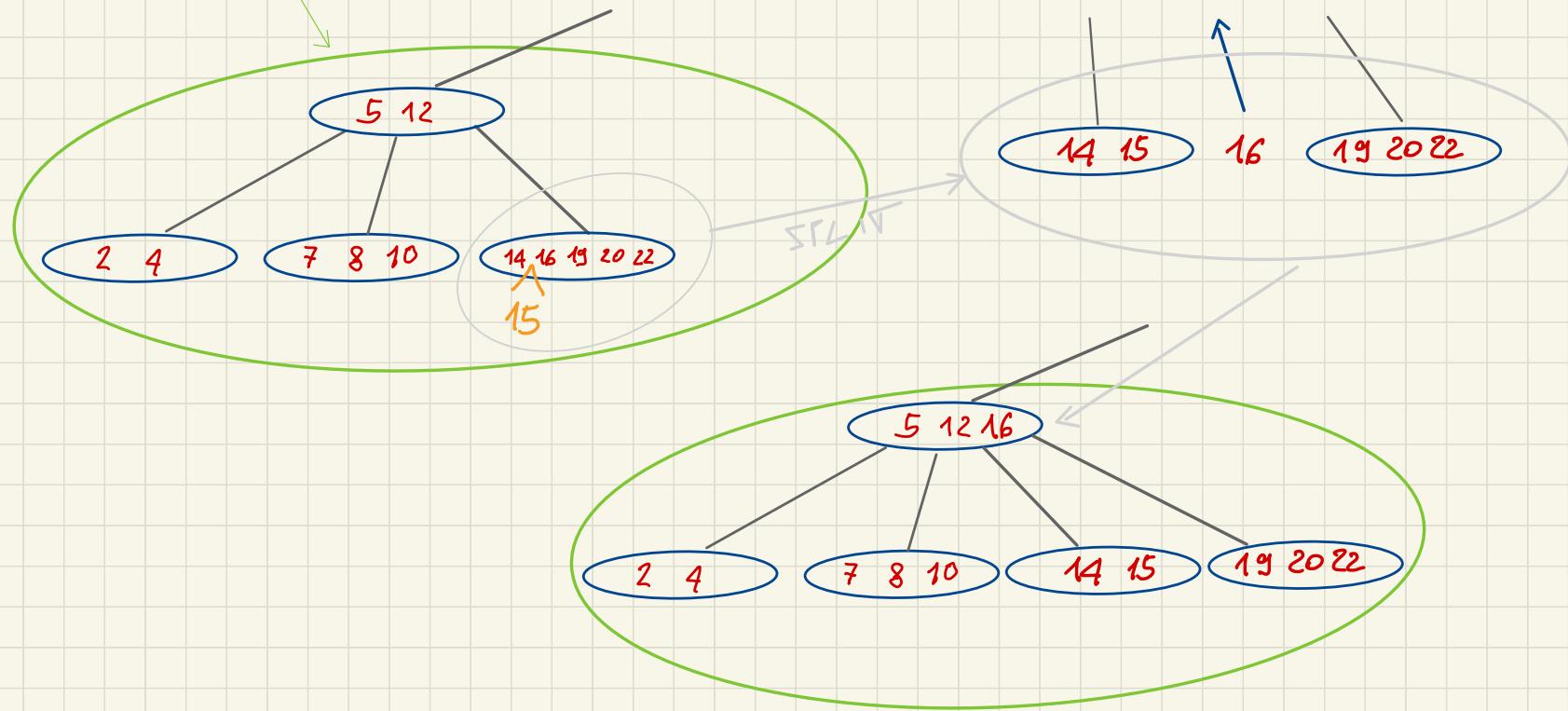
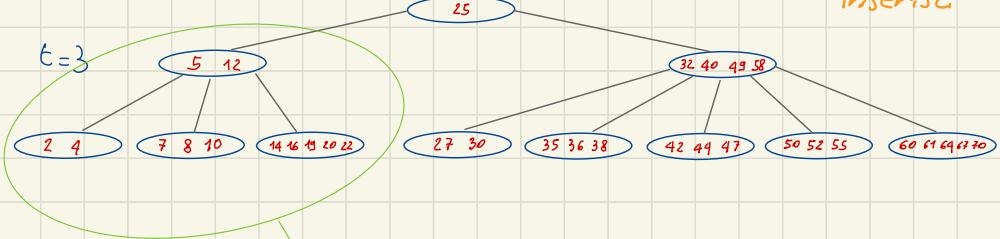


Inserimento

- Ricerca la foglia in cui l'elemento va inserito



inserisci 15



B-alberi: rieplogo costi

$n = n^o$ di chiavi

t

per ogni chiave

accessi a memoria
di massa

• RICERCATI

$\Theta(\lg n)$

$\approx \lg t^n$

• INSERIMENTO

CANCELLAZIONE

$\Theta(t \cdot \lg n)$

$\approx C \cdot \lg t^n$

C costante piccola

B-alberi : riepilogo costo operazioni

($n = n^{\circ}$ chiavi)

passi di calcolo
(tempo)

accessi a
memoria di
massa

• RICERCA

$$\Theta(\lg n)$$

$$\approx \lg_e n$$

• INSERIMENTO

• CANCELLAZIONE

$$\left. \begin{array}{l} \text{• INSERIMENTO} \\ \text{• CANCELLAZIONE} \end{array} \right\} \Theta(t \cdot \lg n)$$

$$\approx C \cdot \lg_e n$$

C costante piccola
(dipende dall'implementazione)
indicativamente $C \approx 4$

COSTI OPERAZIONI su dizionari

di confronti in funzione del numero n di chiavi

	Array ordinati	Alberi binari di ricerca	Alberi AVL Alberi 2-3
Ricerca	$\Theta(\log n)$	$\Theta(n)$	$\Theta(\log n)$
Inserimento	$\Theta(n)$	$\Theta(n)$	$\Theta(\log n)$
Cancellazione	$\Theta(n)$	$\Theta(n)$	$\Theta(\log n)$

Svantaggio
Capacità fissa

Custo per i puntatori

Tabelle Hash

TABELLE HASH : idea generale

- DIZIONARIO memorizzato in un array (tabella)
- La POSIZIONE di un elemento viene ottenuta applicando una funzione alla chiave



funzione hash

chiave \xrightarrow{h} posizione

0	
1	
2	
3	
:	
$m-1$	

FUNZIONI HASH

$U =$ universo delle chiavi

$\{0, \dots, m-1\}$ spazio degli indici

FUNZIONE HASH

$$h: U \rightarrow \{0, \dots, m-1\}$$

chiave \xrightarrow{h} posizione

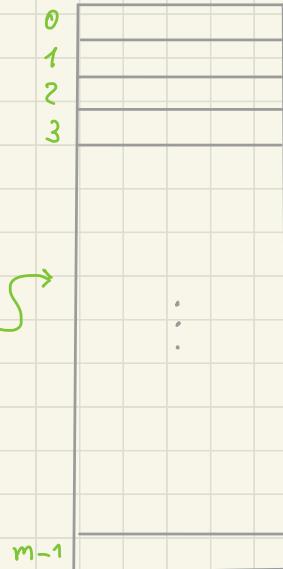


TABELLE AD ACCESSO DIRETTO : un caso semplice

- Chiavi univoche intere nel range $0..m-1$ $\cup = \{0, \dots, m-1\}$

\Rightarrow ARRAY di m posizioni
indice \leftrightarrow chiave



RICERCA
INSERIMENTO
CANCELLAZIONE

} Tempo $O(1)$

Studenti \rightarrow n° matricola n° nel range 0..999999

\downarrow
n studenti

che seguono obiettivo ≤ 200

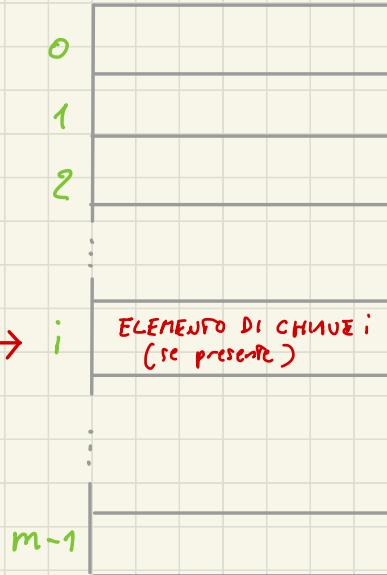
$$U = \{0, \dots, 999999\}$$

$$m = 1000000$$

$$n = 200$$

$$\frac{n}{m} = \frac{200}{1.000.000} = 0.02\%$$

$i \longrightarrow i$ ELEMENTO DI CHIAVE
(se presente)



FATTORE DI CARICO α

$$\alpha = \frac{n}{m}$$

quando la tabella è piena

$n \ll m \rightarrow \alpha \approx \frac{1}{m}$

$\alpha = 1$ tabella piena

$\alpha = 0$ tabella vuota

$$U = \{ 950.000, \dots, 999.999 \}$$

$$m = 50000$$

$$h(x) \approx x - 950.000 \quad \text{fz. harsh}$$

$$\alpha = \frac{n}{m} = \frac{200}{50000} = \frac{2}{500} = 0.4\% \quad \text{specio}$$

FUNZIONI HASH

U = universo delle chiavi

$\{0, \dots, m-1\}$ spazio degli indici

FUNZIONE HASH

$h: U \rightarrow \{0, \dots, m-1\}$

trasformazione di chiavi
in indici (posizioni)

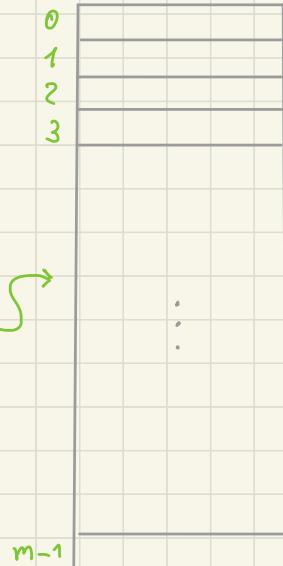
$m = 1000$

$U = \{950, 000, \dots, 999, 999\}$

$$h(x) = x \bmod 1000$$

$$\alpha = \frac{200}{1000} = \frac{1}{5} = 20\%$$

chiave \xrightarrow{h} posizione



FUNZIONI HASH PERFETTE

$h: U \rightarrow \{0, \dots, m-1\}$ è PERFETTA

Se $u \neq v \Rightarrow h(u) \neq h(v)$ \equiv iniezione