Lezione 24– B-tree: caratteristiche ed esercizi

Prof.ssa Maria De Marsico
demarsico@di.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

Parte del materiale è tratto dal materiale preparato dalla prof.ssa Moscarini per il canale A-L

Dati che ammettono un ordinamento significativo per l’applicazione

- Il fatto di avere un ordinamento delle chiavi ci ha dato buoni risultati ed ha migliorato le prestazioni delle operazioni in termini di numero di accessi a memoria.

- Il B-tree nasce dalla **generalizzazione** della struttura di indice.

Ordine lessicografico

SAPIENZA
Università di Roma
Dipartimento di Informatica

-

Un *alfabeto finito totalmente ordinato* di simboli è un insieme $\Sigma = (\delta_1, \delta_2, \dots \delta_n)$, … dotato di un ordine totale . $\delta_1 < \delta_2 < \dots < \delta_n$

Date due sequenze di simboli

$$I = \delta_{i1}\, \delta_{i2}\, \dots \delta_{in}$$
$$J = \delta_{j1}\, \delta_{j2} \dots \delta_{jm}$$

diciamo che ***I<J*** se esiste un numero $k \in \mathbb{N}$ per cui

$\delta_{i1}\, \delta_{i2}\, \dots \delta_{ik} = \delta_{j1}\, \delta_{j2} \dots \delta_{jk}$

e vale **una** delle seguenti relazioni:

$\delta_{i(k+1)} = \delta_{j(k+1)}$ oppure $n = k < m$

.

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Algoritmo di confronto (banale ...)

La regola data sopra è equivalente al seguente algoritmo di confronto:

- si pone n=1
- si **confrontano** i simboli nella **posizione n-esima** della stringa:
  - se una delle due stringhe **non possiede** l'elemento n-esimo, **allora è minore** dell'altra e l'algoritmo termina
  - se entrambe le stringhe **non possiedono** l'elemento n-esimo, allora sono **uguali** e l'algoritmo termina
  - se i simboli **sono uguali**, si passa alla posizione successiva della stringa (n=n+1)
  - se questi sono **diversi**, il loro **ordine** è **l'ordine delle stringhe**

E’ una generalizzazione del file con indice.

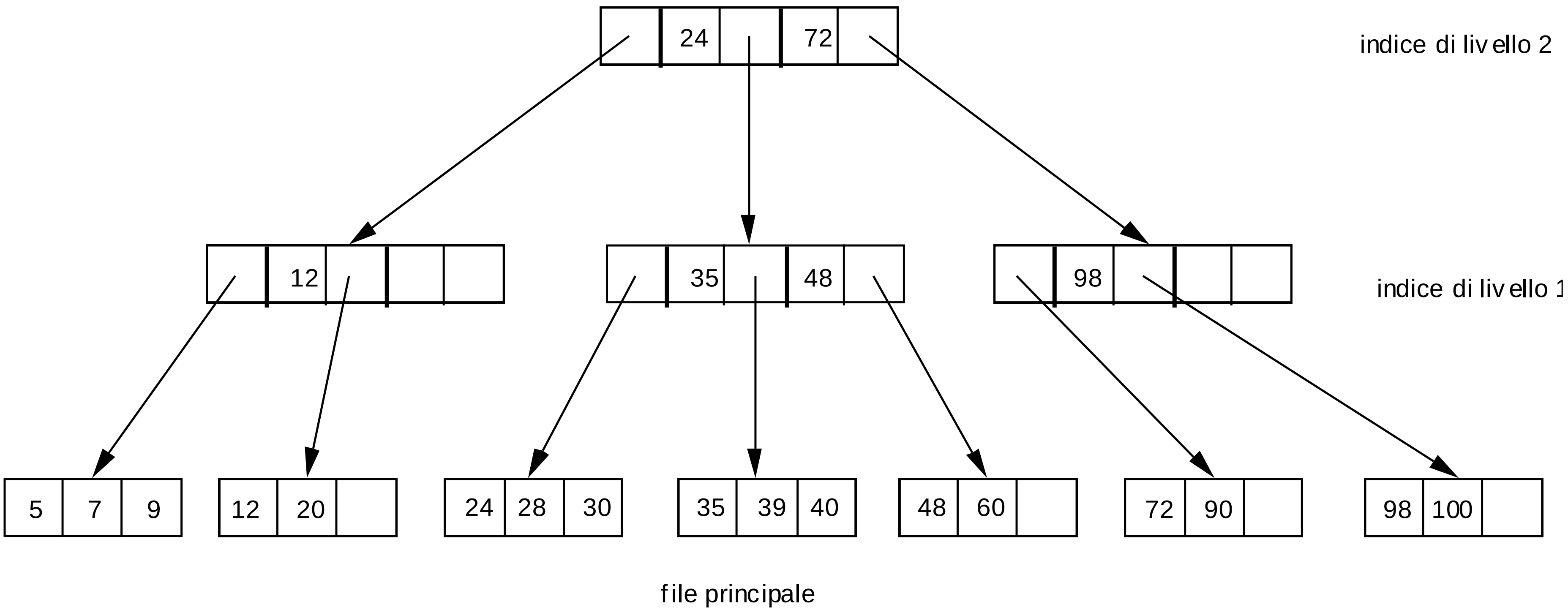
si accede al file attraverso una **gerarchia** di indici

l’indice a **livello più alto** nella gerarchia (la **radice**) è costituito da un **unico blocco** e quindi **può risiedere in memoria principale** durante l’utilizzo del file.

B-tree

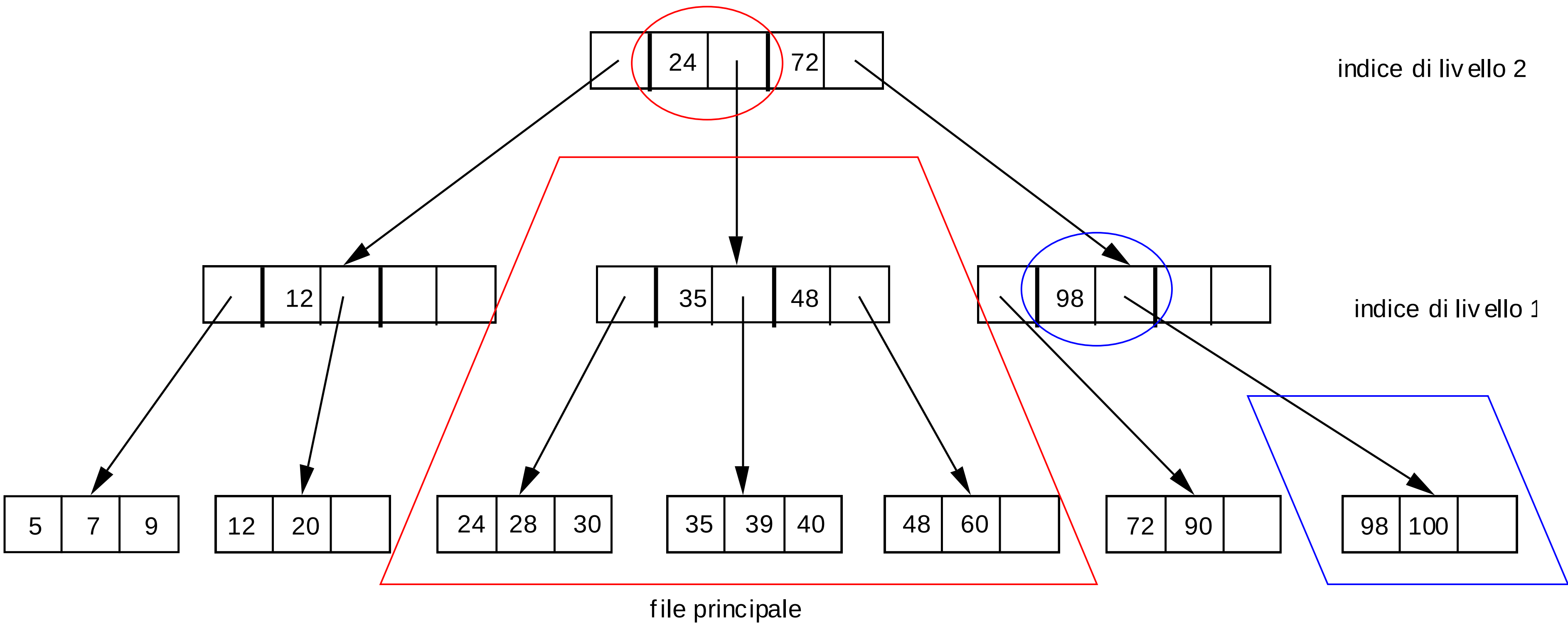
Sapienza
Università di Roma
Dipartimento di Informatica

- Ogni blocco di un file **indice** è costituito di record contenenti una **coppia (v,b)** dove **v è il valore della chiave** del **primo** record **della porzione del file principale che è accessibile attraverso il puntatore b**; b può essere un puntatore ad un blocco del file indice a livello **immediatamente più basso** oppure (nei record del file indice nel più basso livello della gerarchia) ad un blocco del **file principale.**
- Il **primo record indice** di ogni blocco indice contiene **solo** un **puntatore** ad un blocco le cui chiavi sono **minori** di quelle nel blocco puntato dal secondo record indice
- Un blocco del file **principale** è memorizzato come quello di un ISAM
- **Ogni** blocco di un B-tree (indice o file principale) deve essere **pieno almeno per metà** (della taglia!) tranne eventualmente la **radice (più importante che sia un unico blocco)**


24
72
indice di livello 2
12
35
48
98
indice di livello 1
5
7
9
12
20
24
28
30
35
39
40
48
60
72
90
98
100
file principale

B-tree

- **Nota:** ogni record indice ha una chiave che **ricopre** quelle del **sottoalbero** che parte dal blocco **puntato**


24
72
indice di livello 2
12
35
48
98
indice di livello 1
5
7
9
12
20
24
28
30
35
39
40
48
60
72
90
98
100
file principale

Ricerca

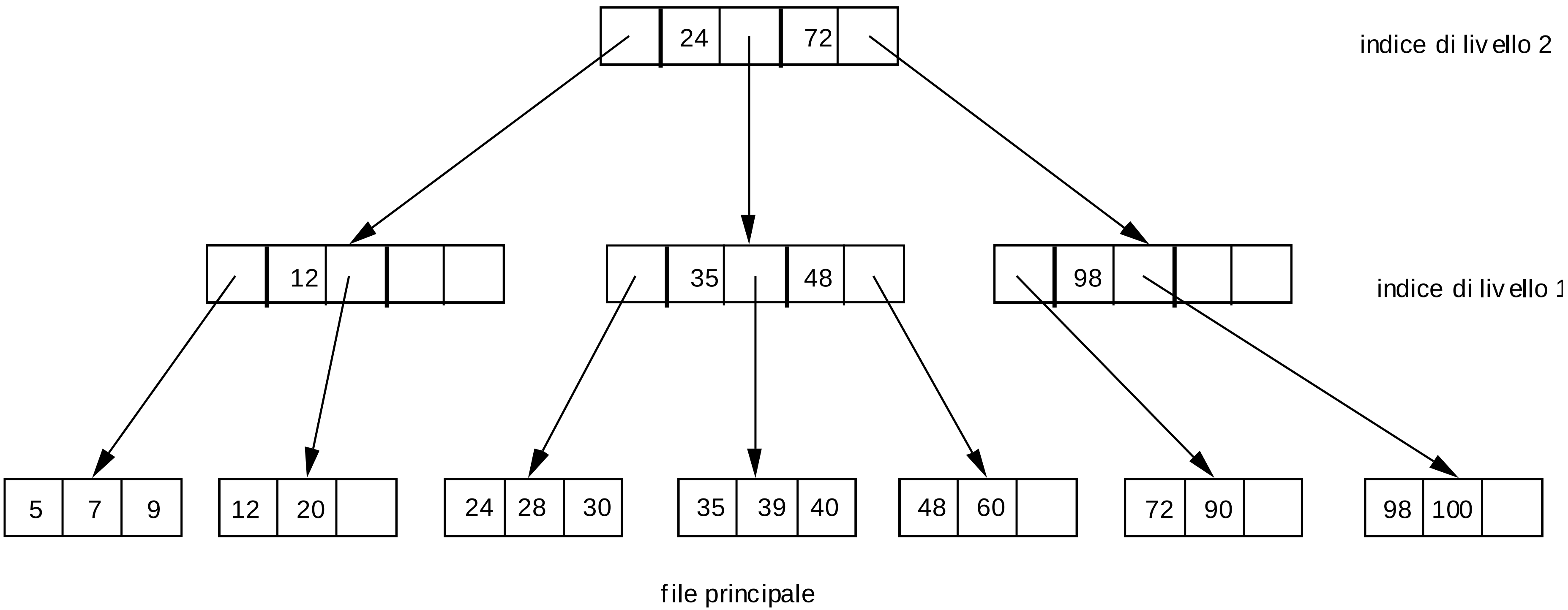
SAPIENZA
Università di Roma
Dipartimento di Informatica

- Durante la ricerca di un record con un dato valore per la chiave si accede agli indici a **partire da quello a livello più alto**; a mano a mano che si **scende** nella gerarchia di indici **si restringe la porzione** (insieme di blocchi) del file principale **in cui deve trovarsi il record desiderato**, fino a che, **nell'ultimo livello** (il più basso nella gerarchia) tale porzione **è ristretta ad un unico blocco**.

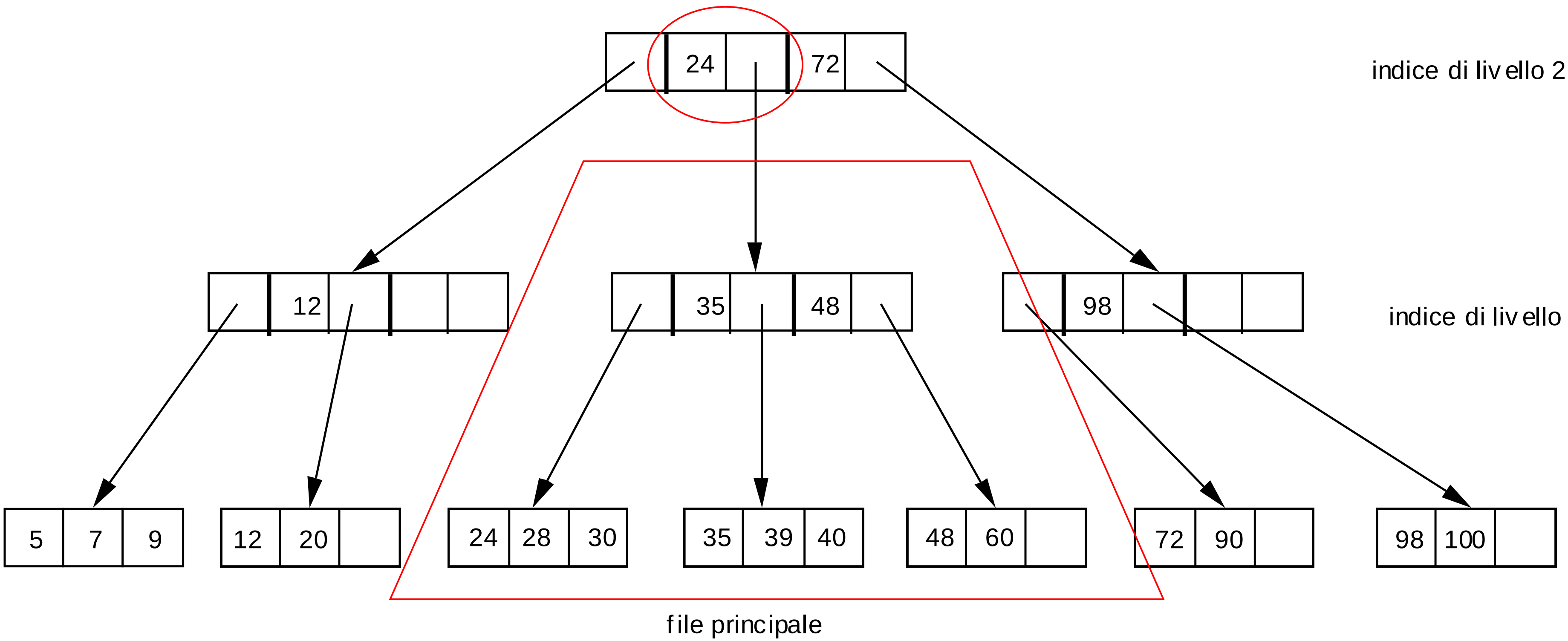
  Più in dettaglio, per ricercare il record del file principale con un dato valore v per la chiave si procede nel modo seguente. Si parte **dall'indice a livello più alto** (che è costituito da un **unico** blocco) e **ad ogni passo si esamina un <u>unico</u> blocco**. Se il blocco esaminato è un blocco del file **principale**, tale blocco è quello **in cui deve** trovarsi il record desiderato; se, invece, è un blocco di un file **indice**, si cerca in tale blocco **un valore della chiave che ricopre v** e si **segue il puntatore associato** (che sarà o un puntatore ad un blocco dell'indice al livello immediatamente inferiore o un blocco del file principale).

Ricerca

Sapienza Università di Roma
Dipartimento di Informatica

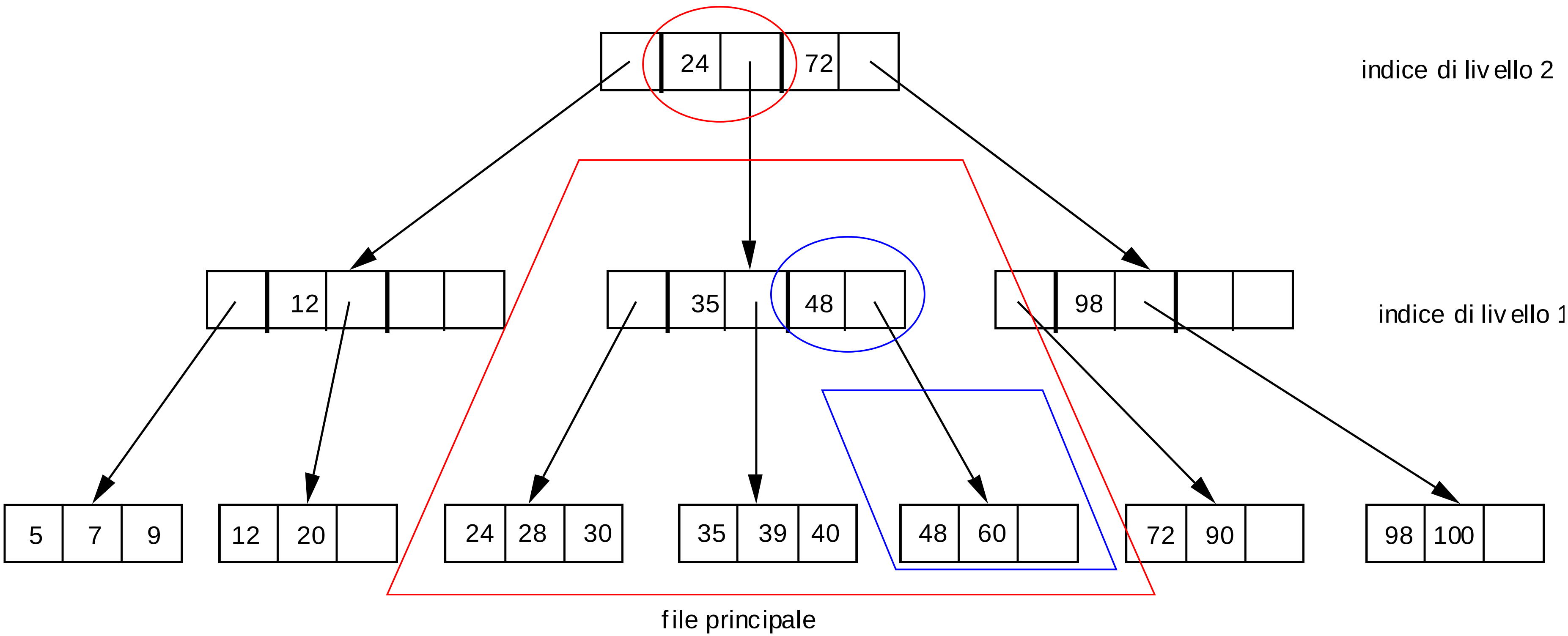


Ricerca


24
72
indice di livello 2
12
35
48
98
indice di livello 1
5 7 9
12 20
24 28 30
35 39 40
48 60
72 90
98 100
file principale

Ricerca

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA



SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Per la ricerca sono necessari

$h+1$ accessi

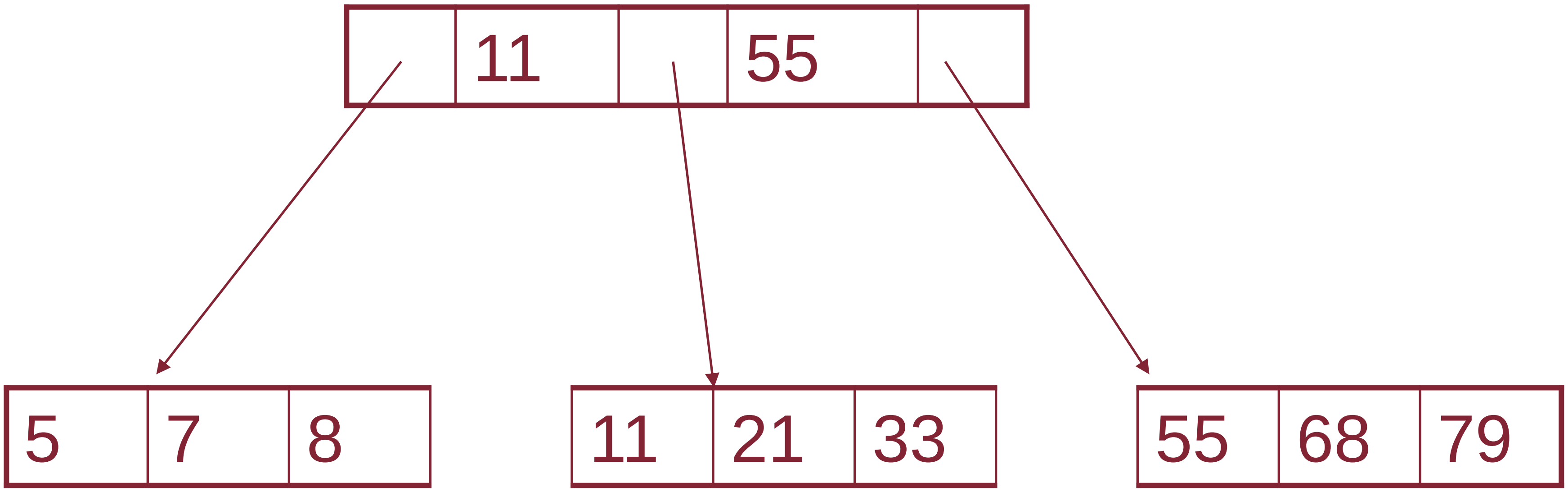
dove h è l’altezza dell’albero

Osservazione

- **Più sono pieni** i blocchi **più h è piccolo** (e quindi **meno costa** la ricerca)

- Conseguenza: si richiede che ogni blocco (sia del file principale che del file indice) sia pieno almeno per metà

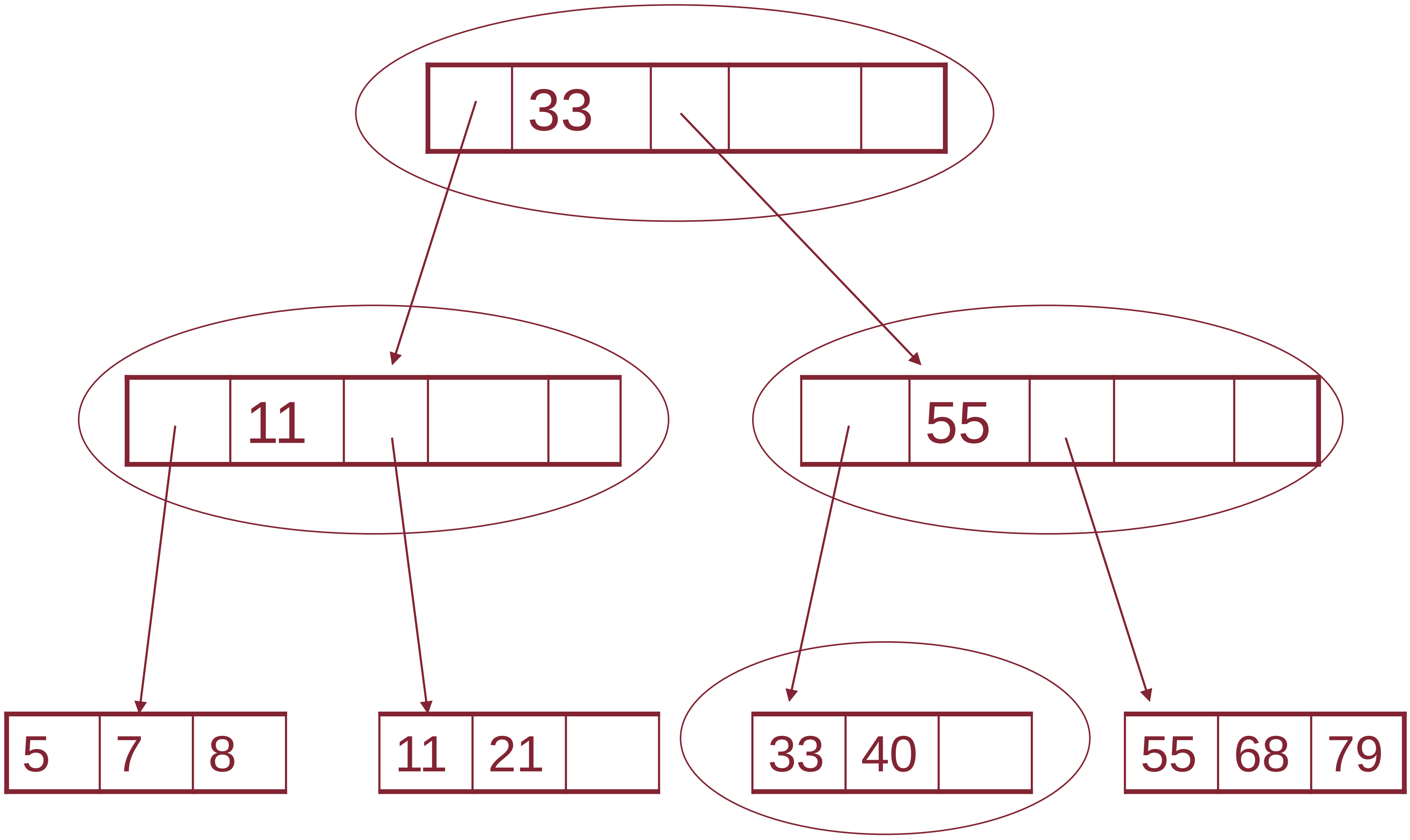
- Se i blocchi **sono completamente pieni** un **inserimento <u>può</u>** richiedere **una modifica dell'indice ad ogni livello** e in ultima ipotesi **può far crescere l'altezza dell'albero di un livello**



Si vuole inserire il record con chiave 40

Ogni blocco del file principale deve contenere **almeno** 2 record

Supponiamo lo stesso per l’indice (poco realistico ..)



33
11
55
5
7
8
11
21
33
40
55
68
79

Qual è il massimo valore k che può assumere h (qual è la profondità dell'albero nel caso **peggiore**)?

Massimo valore di h

- Siano:
- N numero di record nel file principale
- 2e-1 numero di record del file principale che possono essere memorizzati in un blocco
- 2d-1 numero di record del file indice che possono essere memorizzati in un blocco

NOTA: **l'assunzione** che il numero di record del file principale e del file indice che possono essere memorizzati in un blocco **sia dispari** viene fatta **esclusivamente per rendere semplici i calcoli**

Massimo valore di h

Poiché i blocchi devono essere pieni **almeno** per metà:

- ogni blocco del file principale deve contenere **almeno e** record
- ogni blocco del file indice deve contenere **almeno d** record

Massimo valore di h

L'altezza massima dell'albero denotata con **k** si ha quando i blocchi sono pieni al minimo, cioè quando

- ogni blocco del file principale contiene **esattamente e** record
- ogni blocco del file indice contiene **esattamente d** record

Massimo valore di h

Pertanto:

- Il file **principale** ha al più **N/e** blocchi
- Al livello 1 il file indice ha **N/e record** che possono essere memorizzati in **N/ed blocchi**
- Al livello 2 il file indice ha **N/ed record** che possono essere memorizzati in **N/ed^2 blocchi**
- …
- Al livello **i** il file indice ha **N/ed^{i-1}** record che possono essere memorizzati in **N/ed^i** blocchi

Massimo valore di h

SAPIENZA
Università di Roma
Dipartimento di Informatica

Al livello k il file indice ha esattamente 1 blocco quindi

$N/ed^k \leq 1$ (ricordiamo che prendiamo la parte intera superiore delle divisioni, inclusa l'ultima, quindi in particolare nell'ultima il risultato non arrotondato è generalmente minore di 1)

$\lceil N/ed^k \rceil = 1$

Consideriamo per semplicità l'uguaglianza (ci interessa una approssimazione sufficientemente vicina al massimo)

$ed^k = N$ da cui $d^k = N/e$

e infine

$$k = \log_d(N/e)$$

Quindi $\log_d(N/e)$ rappresenta un valore che approssima in maniera sufficiente il limite superiore per l'altezza dell'albero

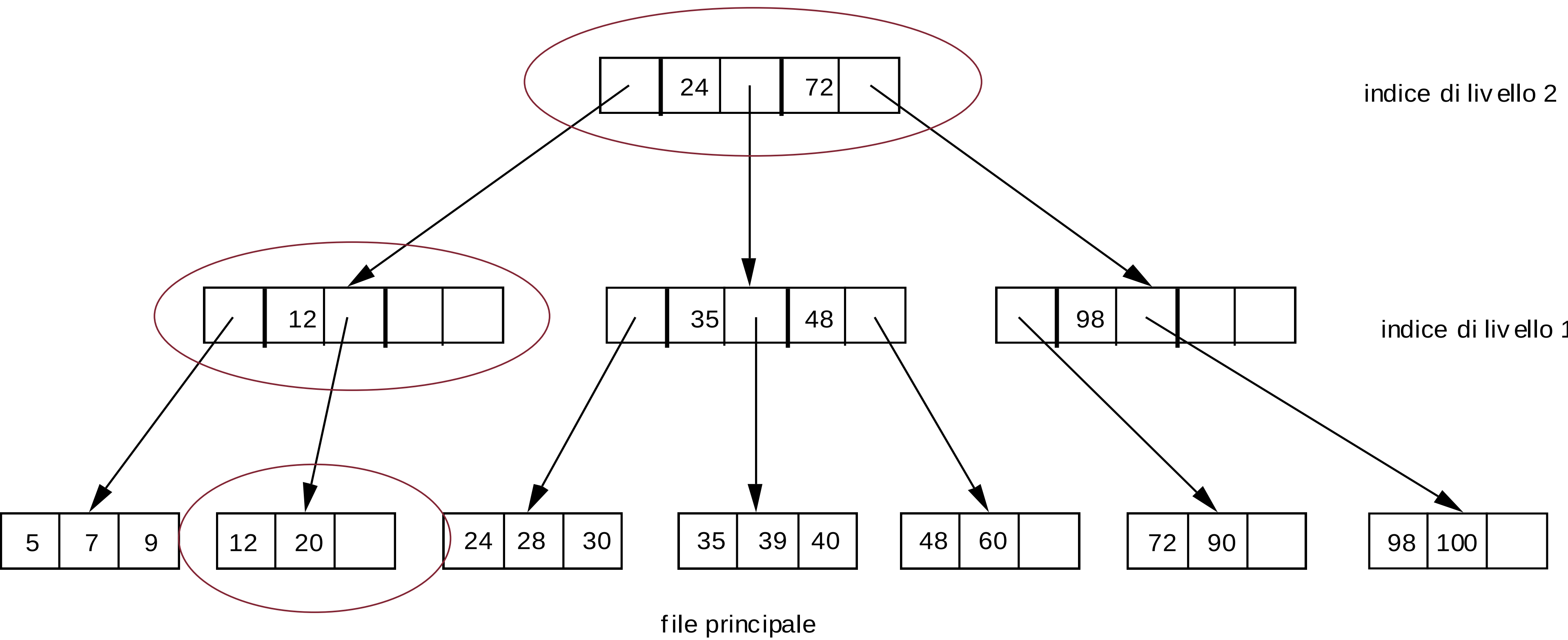
Inserimento

h+1

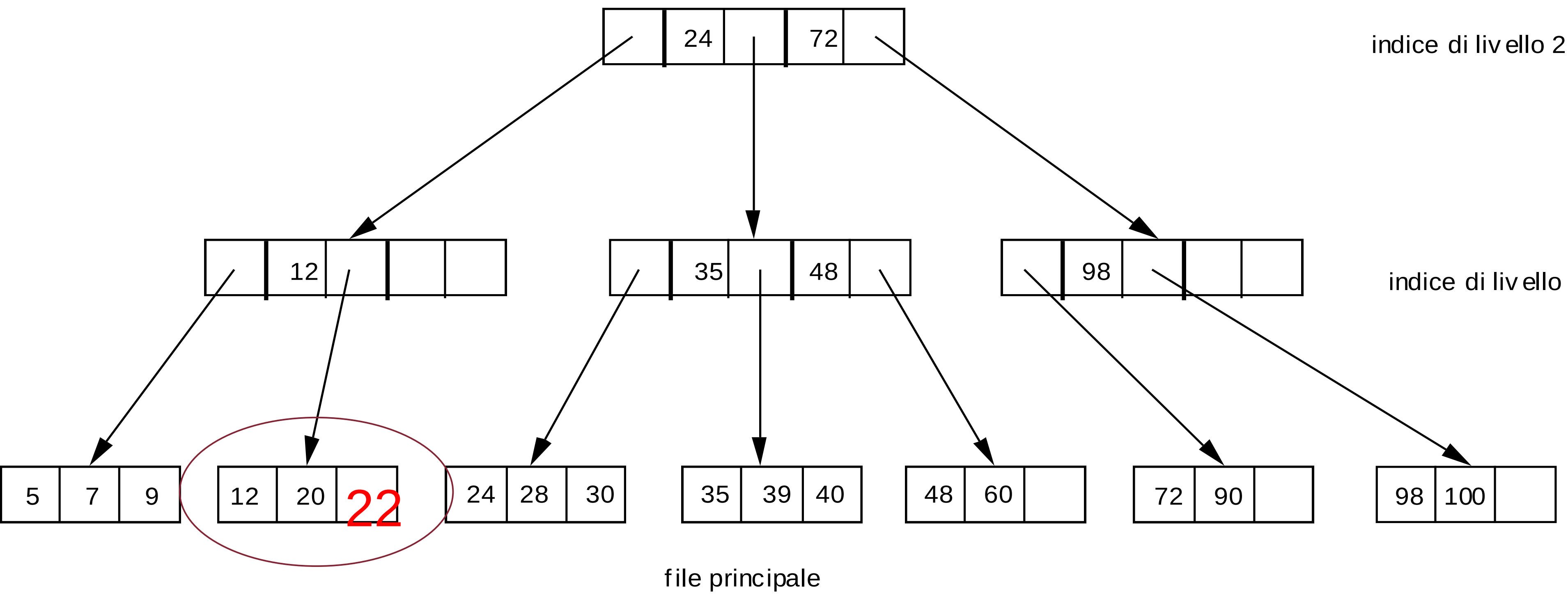
(costo di una ricerca per ricercare il blocco in cui deve essere inserito il record)

+1 accesso per riscrivere il blocco,

se nel blocco **c'è spazio sufficiente** per inserire il record.



si vuole inserire il record con chiave 22



si vuole inserire il record con chiave 22

Inserimento

se nel blocco **non c'è** spazio sufficiente per inserire il record:

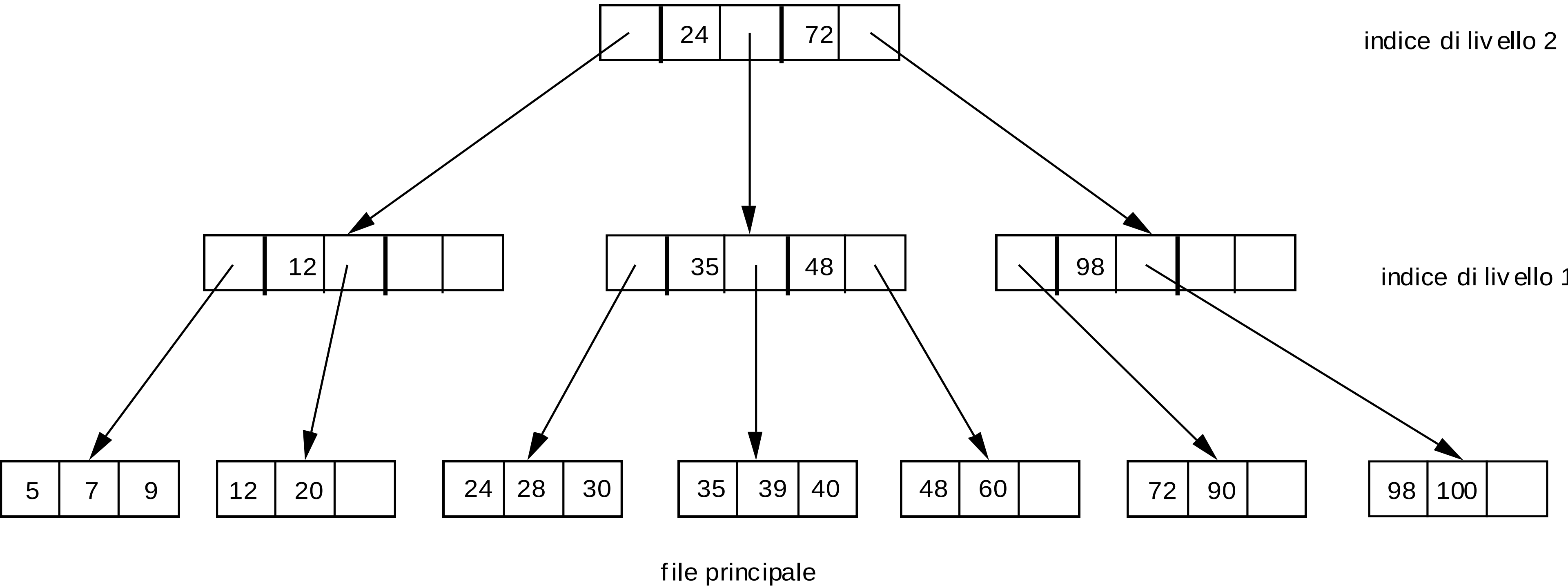
h+1

(costo di una **ricerca** per ricercare il blocco **in cui deve essere inserito** il record)

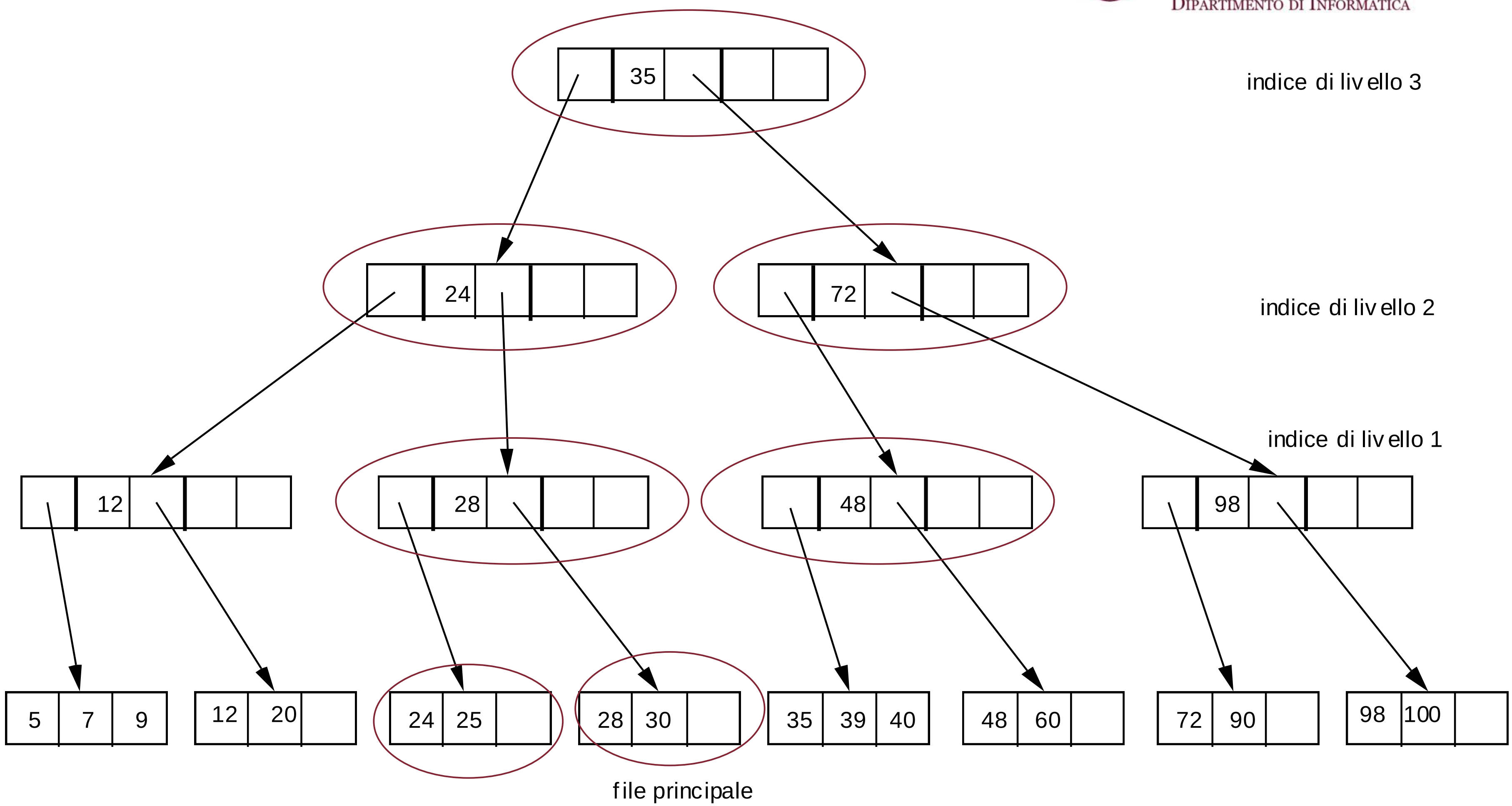
+s accessi (**s≤2h+1**)

(nel caso peggiore per ogni livello dobbiamo sdoppiare un blocco quindi effettuare due accessi più uno alla fine per la nuova radice)

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA



si vuole inserire il record con chiave 25


35
24
72
12
28
48
98
5 7 9
12 20
24 25
28 30
35 39 40
48 60
72 90
98 100
indice di livello 3
indice di livello 2
indice di livello 1
file principale

Cancellazione

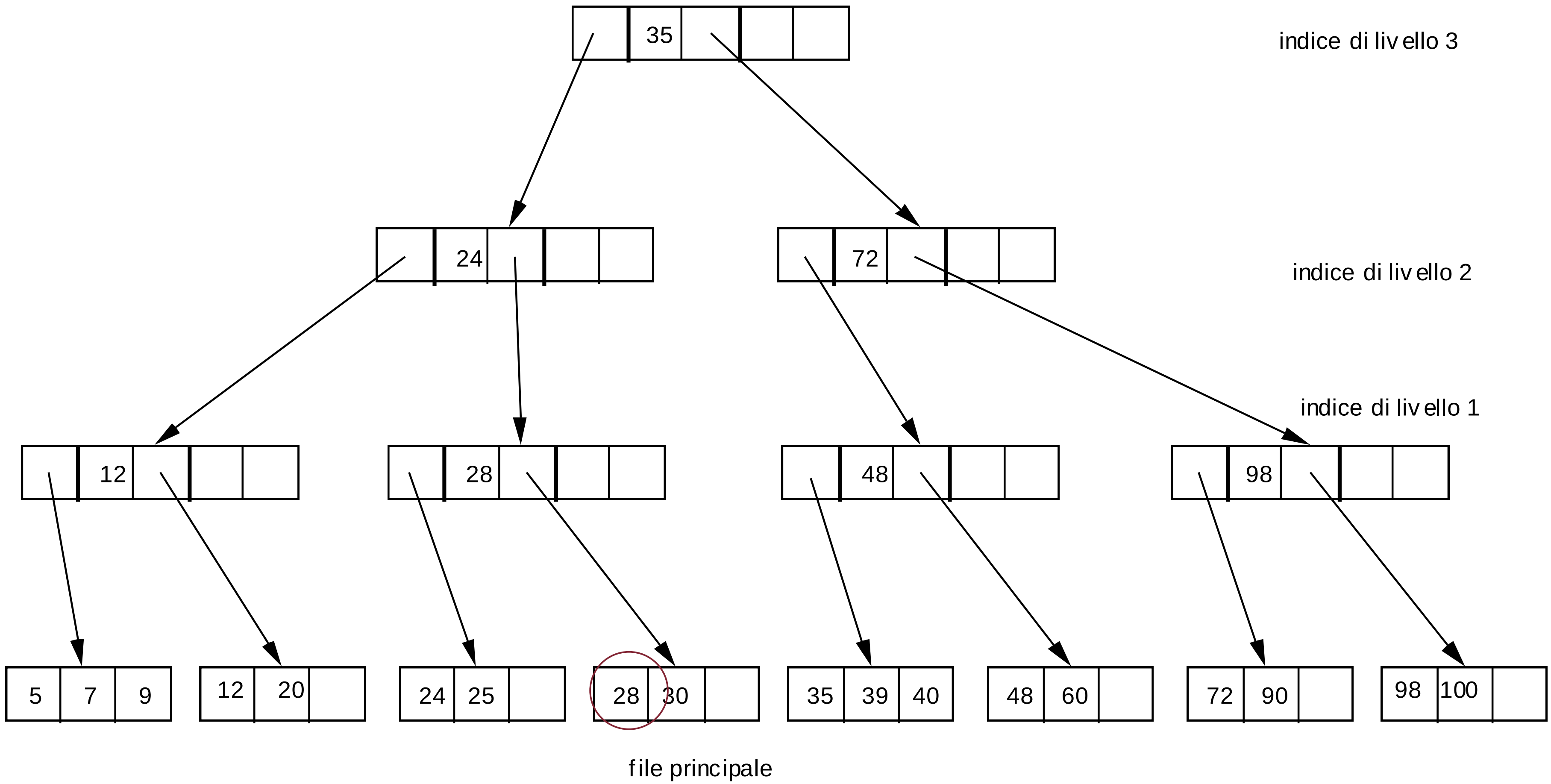
h+1

(costo di una ricerca per ricercare il blocco in cui si trova il record)

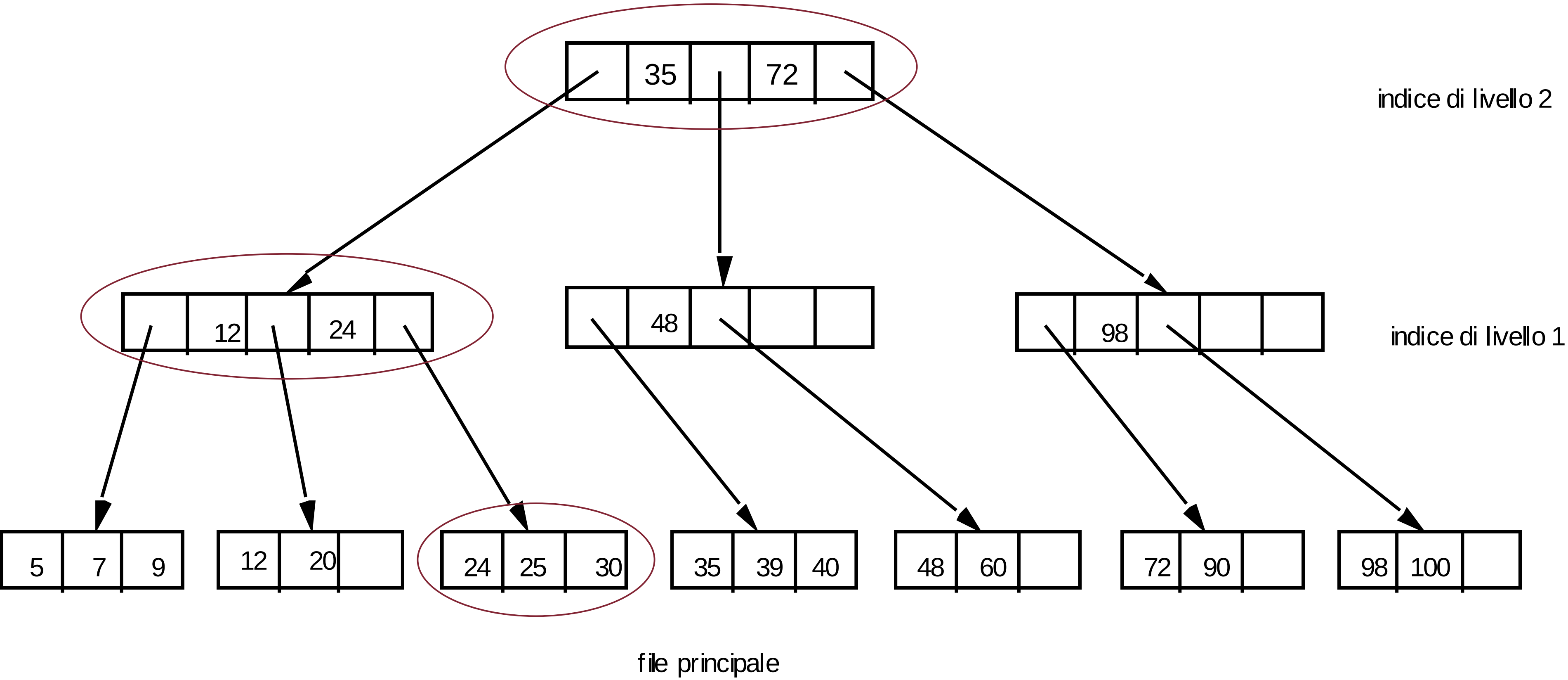
+1 accesso per riscrivere il blocco,

se il blocco rimane **pieno almeno per metà** successivamente alla cancellazione,

altrimenti sono necessari ulteriori accessi



Si vuole cancellare il record con chiave 28


35
72
indice di livello 2
12
24
48
98
indice di livello 1
5
7
9
12
20
24
25
30
35
39
40
48
60
72
90
98
100
file principale

Modifica

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

h+1

(costo di una ricerca per ricercare il blocco in cui si trova il record)

+1 accesso per riscrivere il blocco,

se la modifica non coinvolge campi della chiave,

altrimenti :

costo cancellaz. + costo inserimento

Osservazioni

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

In un B-tree per definizione ogni blocco è sempre pieno **almeno** per metà (si intende metà **dello spazio fisico, cioè dei byte**), quindi bisognerebbe **verificare** che questa condizione sia **sempre** soddisfatta.

Per quanto riguarda il **file principale**

se il numero di record massimo che possiamo memorizzare è **dispari**, quindi esprimibile nella forma 2E-1, possiamo **direttamente** considerare **E** come occupazione **minima**,

se il numero massimo di record che possiamo memorizzare è **pari**, quindi esprimibile come 2E, possiamo considerare **E+1** come occupazione **minima**, **tranne il caso particolare in cui la taglia del record è esattamente un sottomultiplo** di quella del blocco e la metà dei record riempie esattamente la metà dei byte.

Per sicurezza … verificare.

Per il file indice, valgono grosso modo le stesse considerazioni, ma occorre fare attenzione al fatto che, dei record che memorizziamo in un blocco, il **primo** contiene **sempre solo** un puntatore, quindi i calcoli vanno eseguiti di conseguenza.

Esempio 1

Supponiamo di avere un file di 170.000 record. Ogni record occupa 200 byte, di cui 20 per il campo chiave. Ogni blocco contiene 1024 byte. Un puntatore a blocco occupa 4 byte.

Se usiamo un B-tree e assumiamo che sia i blocchi indice che i blocchi del file sono pieni al **minimo**, quanti blocchi vengono usati per il livello foglia (file principale) e quanti per l'indice, considerando tutti i livelli non foglia ?

Qual è il costo di una ricerca in questo caso?

Esempio 1

Sapienza Università di Roma
Dipartimento di Informatica

Abbiamo i seguenti dati:

- il file contiene 170.000 record: NR = 170.000
- ogni record occupa 200 byte: R = 200
- il campo chiave occupa 20 byte: K = 20
- ogni blocco contiene 1024 byte: CB = 1024
- un puntatore a blocco occupa 4 byte: P = 4

Esempio 1

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Per quanto riguarda i blocchi di dati nel livello **foglia** (file principale) indichiamo il numero massimo di record memorizzabili con MR.

Avremo

- MR = $\lfloor$ CB/R $\rfloor$ = $\lfloor$ 1024/200 $\rfloor$ = 5. Poiché il numero in questo caso è dispari (MR = 2E-1 = 5) per definizione, un blocco dati non conterrà mai meno di E = 3 record, e comunque non può contenerne più di 2E-1 = 5. L'esercizio indica che i blocchi sono pieni al **minimo**, quindi ogni record contiene E record. A questo punto possiamo calcolare il numero di blocchi nel livello foglia, che sarà BF = $\lceil$NR/E$\rceil$ = $\lceil$170.000/3$\rceil$ = 56.667.

Esempio 1

- Il **primo** livello di indice contiene un record per ognuno dei blocchi del file principale.
- A questo punto dobbiamo calcolare la **capacità dei blocchi indice**. Nei livelli indice del B-tree, **ogni record, eccetto il primo, contiene una coppia (chiave, puntatore)**; il **primo** record invece contiene solo un **puntatore** (relativo al blocco dati che contiene i record con chiave minore della chiave contenuta nel record indice successivo).
- indicando con MaxI tale numero dovremo avere $(MaxI - 1) \times K + (MaxI) \times P \leq CB$ (ricordiamo che abbiamo **una chiave in meno**), quindi nel nostro caso $(MaxI -1) \times 20 + (MaxI) \times 4 \leq 1024$, cioè esplicitando i passaggi di calcolo
- $(MaxI) \times 20 - 20 + MaxI \times 4 \leq 1024$, $MaxI \times 24 \leq 1044$, $MaxI \leq 1044/24=43{,}5$
- Quindi dovendo essere un numero intero MaxI=43 $2D-1=2 \times 22-1$

Esempio 1

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- In alternativa possiamo sottrarre lo spazio richiesto dal record con il solo puntatore dalla capacità del blocco, e calcolare quanti record chiave+puntatore sono contenuti nei byte che rimangono**. Bisogna però ricordare di riaggiungere 1 al numero di record complessivi, perché anche il record con il solo puntatore è a tutti gli effetti un record indice.** Nel nostro caso
- MaxI = $\lfloor(1024 - 4)/24\rfloor+1 = \lfloor 1020/24 \rfloor +1 = \lfloor 42,5 \rfloor +1 = 42 + 1 = 43$

- **Verifichiamo però** che la metà dei record riempia la metà del blocco, considerando che il primo è solo puntatore
- $21 \times(20+4)+4= 508$ (512) che è meno della metà del blocco quindi per l'occupazione minima aggiungiamo 1 record (arriviamo a occupare 532).

- Occupazione minima MinI=23 record indice per blocco indice

Esempio 1

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Al **primo** livello di indice avremo quindi B1 = $\lceil BF/MinI \rceil$ = $\lceil 56.667/23 \rceil$ = $\lceil 2463{,}78 \rceil$ = 2464 blocchi indice.
- Ad ognuno di questi corrisponderà un record al **secondo** livello di indice, quindi B2 = $\lceil B1/MinI \rceil$ = $\lceil 2464/23 \rceil$ = $\lceil 107{,}13 \rceil$ = 108 blocchi indice. Iterando il ragionamento avremo B3 = $\lceil B2/MinI \rceil$ = $\lceil 108/23 \rceil$ = 5 blocchi indice. A questo punto **attenzione**: anche questi ultimi **sono blocchi** a cui devono corrispondere **dei record indice al livello superiore**. Arriviamo così alla radice, che deve contenere sempre un **unico** blocco, e per la quale si rilascia il vincolo che sia piena almeno per metà (infatti nel nostro caso conterrà solo 5 record). Abbiamo allora B4 = 1 blocco indice.
- In conclusione abbiamo BF = 56.667 blocchi dati al livello foglia, e complessivamente BI = B1 + B2 + B3 + B4 = 2464 + 108 + 5 + 1 = 2578 blocchi nei 4 livelli di indice;
- **Il costo di una ricerca sarà di 5 accessi** (un blocco per ognuno dei 4 livelli di indice + 1 blocco del il file principale)

Esempio 2

Supponiamo di avere un file di 170.000 record. Ogni record occupa 200 byte, di cui 20 per il campo chiave. Ogni blocco contiene 1024 byte. Un puntatore a blocco occupa 4 byte.

Se usiamo un B-tree e assumiamo che sia i blocchi indice che i blocchi del file sono pieni al **massimo**, quanti blocchi vengono usati per il livello foglia (file principale) e quanti per l'indice, considerando tutti i livelli non foglia ?

Qual è il costo di una ricerca in questo caso?

Esempio 2

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Ricordiamo i dati:

- il file contiene 170.000 record: NR = 170.000
- ogni record occupa 200 byte: R = 200
- il campo chiave occupa 20 byte: K = 20
- ogni blocco contiene 1024 byte: CB = 1024
- un puntatore a blocco occupa 4 byte: P = 4

Esempio 2

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Per quanto riguarda i blocchi di dati nel livello **foglia** (file principale) indichiamo il numero massimo di record memorizzabili con MR.

Abbiamo già calcolato che

- MR = $\lfloor$ CB/R $\rfloor$ = $\lfloor$ 1024/200 $\rfloor$ = 5.
- L'esercizio questa volta indica che i blocchi sono pieni al **massimo**, quindi ogni record contiene 5 record. A questo punto possiamo calcolare il numero di blocchi nel livello foglia, che sarà BF = $\lceil$NR/E$\rceil$ = $\lceil$170.000/5$\rceil$ = 34.000

Esempio 2

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Il **primo** livello di indice contiene un record per ognuno dei blocchi del file principale.

- Abbiamo già calcolato in numero massimo MaxI di record indice in un blocco
- $(\text{MaxI} - 1) \times 20 + (\text{MaxI}) \times 4 \leq 1024$

  oppure
- $\text{MaxI} = \lfloor (1024 - 4)/24 \rfloor + 1 = \lfloor 1020/24 \rfloor + 1 = \lfloor 42{,}5 \rfloor + 1 = 42 + 1 = 43$

Esempio 1

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Al **primo** livello di indice avremo quindi $B1 = \lceil BF/MaxI \rceil = \lceil 34000/43 \rceil = \lceil 790,70 \rceil = 791$ blocchi indice.
- Ad ognuno di questi corrisponderà un record al **secondo** livello di indice, quindi $B2 = \lceil B1/MaxI \rceil = \lceil 791/43 \rceil = \lceil 18,40 \rceil = 19$ blocchi indice. Iterando il ragionamento avremo $B3 = \lceil B2/MaxI \rceil = \lceil 19/43 \rceil = 1$ blocco quindi siamo arrivati alla radice.
- In conclusione abbiamo BF = 34000 blocchi dati al livello foglia, e complessivamente BI = B1 + B2 + B3 = 791+ 19 + 1 = 811 blocchi nei 3 livelli di indice;
- **Il costo di una ricerca sarà di 4 accessi** (un blocco per ognuno dei 3 livelli di indice + 1 blocco del il file principale)

- Vediamo come sia l'occupazione totale che il numero di accessi per la ricerca sono diminuiti notevolmente

Esempio 3

Supponiamo di avere un file di 170.000 record. Ogni record occupa 200 byte, di cui 20 per il campo chiave. Ogni blocco contiene 1024 byte. Un puntatore a blocco occupa 4 byte.

- **Se usiamo un B-tree quale sarà il costo massimo di una ricerca?**
- I calcoli da fare corrispondono al caso di numero minimo di record per blocco (record pieni a metà) che comporta il massimo dell'occupazione e dell'altezza dell'albero

- **Se usiamo un B-tree quale sarà il costo minimo di una ricerca?**
- I calcoli da fare corrispondono al caso di numero massimo di record per blocco (record pieni per intero) che comporta il minimo dell'occupazione e dell'altezza dell'albero