B+Trees

Introduzione

B+Trees

- Il B+Trees e’ la variante maggiormente utilizzata dei BTrees
- BTrees e B+trees fanno parte della famiglia degli alberi di ricerca.
- Nel B+Trees i dati sono memorizzati solo nelle foglie. I nodi interni ospitano solo delle chiavi per le operazioni di ricerca
- I B+Trees non avendo dati nei nodi interni sono particolarmente indicati per l’uso nei DataBase
  - Configurazione ideale: tutti i nodi interni matenuti in memoria centrale e foglie su memoria di massa
  - In realta’ si riescono a mantenere in memoria centrale solo i primi livelli dei nodi interni

B+Trees: alcune proprieta’

- Un B+Tree di ordine *m* e’ un albero in cui ogni nodo interno puo’ avere al massimo *m* figli e in cui si possono memorizzare al massimo m-1 key value
- I nodi foglia si trovano tutti allo stesso livello
- I nodi interni memorizzano solo i valori delle chiavi necessari per guidare le ricerche
- Le informazioni (chiave piu’ contenuto del record oppure chiave piu’ puntatore al contenuto del record) sono memorizzate solo nelle foglie

Nodo: proprieta’

- Un nodo di un B+Tree di ordine m:
  - E’ o un nodo interno o una foglia
  - La radice e’ o una foglia o ha **almeno** 2 figli
  - Ogni nodo, ad eccezione della radice e delle foglie, deve essere half full → cioe’ deve avere un numero di figli compreso tra m/2 ed m.
  - Le foglie occupano tutte lo stesso livello

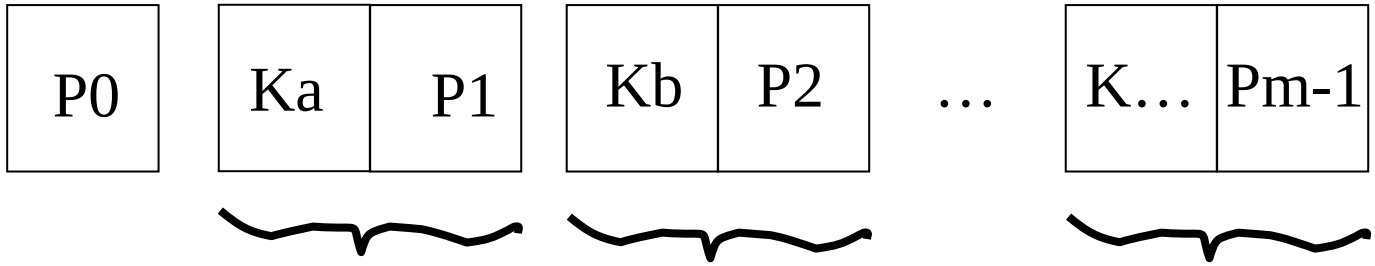
Radice: proprieta’

- E’ una foglia quando e’ l’unico nodo dell’albero, in questo caso deve contenere almeno un record di informazioni (o un puntatore a record di informazioni)
- Quando e’ un nodo interno invece **deve** avere **almeno** due figli

Nodi interni: proprieta’

- Un nodo interno di un B+Tree di ordine m:
  - Puo’ ospitare fino a m puntatori a nodi figli e fino a m-1 key
- Deve essere sempre mezzo pieno (half full)
  - Deve avere almeno m/2 figli
    - Se m=5, m/2 vale 2 o 3? R: a libera scelta, purche’ dentro lo stesso albero si rimanga coerenti. Nella pratica si cerca di mantenere i nodi piu’ pieni possibile.
  - Unica eccezione all’half full requirement: i figli del nodo radice.
    - Prende il sopravvento la regola che dice che la radice deve avere almeno due figli.
    - I figli della radice possono sparire se e solo se le chiavi sono talmente poche che possono essere ospitate tutte nella radice, facendola diventare una foglia

Nodi interni: proprieta’ 2

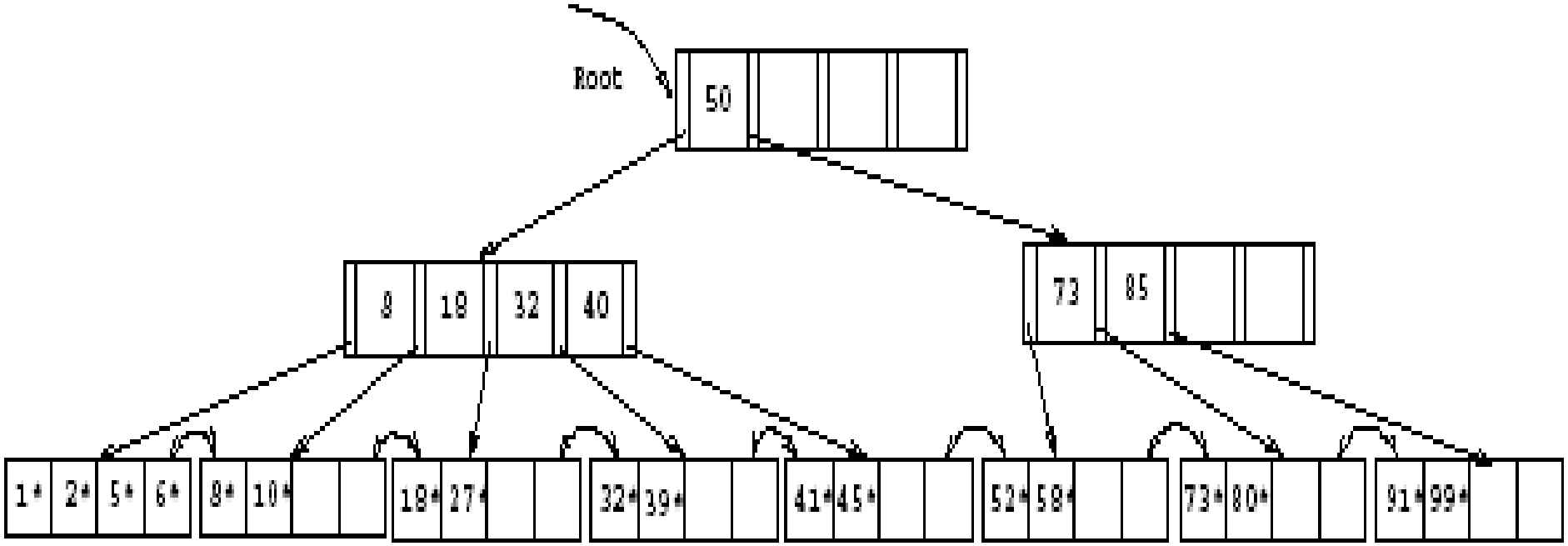


- Ogni chiave ha associato un puntatore (alla sua destra) ad un nodo figlio
  - Nell’esempio, la chiave Kb indica che nel sottoalbero puntato da P2, tutte le chiavi contenute hanno valori maggiori o uguali a Kb
- P0, chiamato anche extra pointer, indica che nel sottoalbero da lui puntato, tutte le chiavi hanno valori strettamente inferiori alla chiave piu’ piccola del nodo (nell’esempio Ka)

Foglie: proprieta’

- Una foglia contiene delle entries costituite da una delle seguenti coppie (nello foglie di uno stesso albero si opta per una sola possibilita’):
  - key e puntatore all’area di allocazione del record
  - key e record dei dati vero e proprio
- Le foglie hanno spazio per ospitare un numero di chiavi che non e’ necessariamente m-1 (puo’ essere maggiore o minore)
  - Un nodo foglia e’ salvato in un blocco fisico del disco. Il numero di key (e record) contenute in un nodo foglia dipende quanti record possono essere ospitati in un blocco fisico
- Le foglie sono collegata l’una all’altra da una doppia lista (puntatore alla precedente e alla successiva)

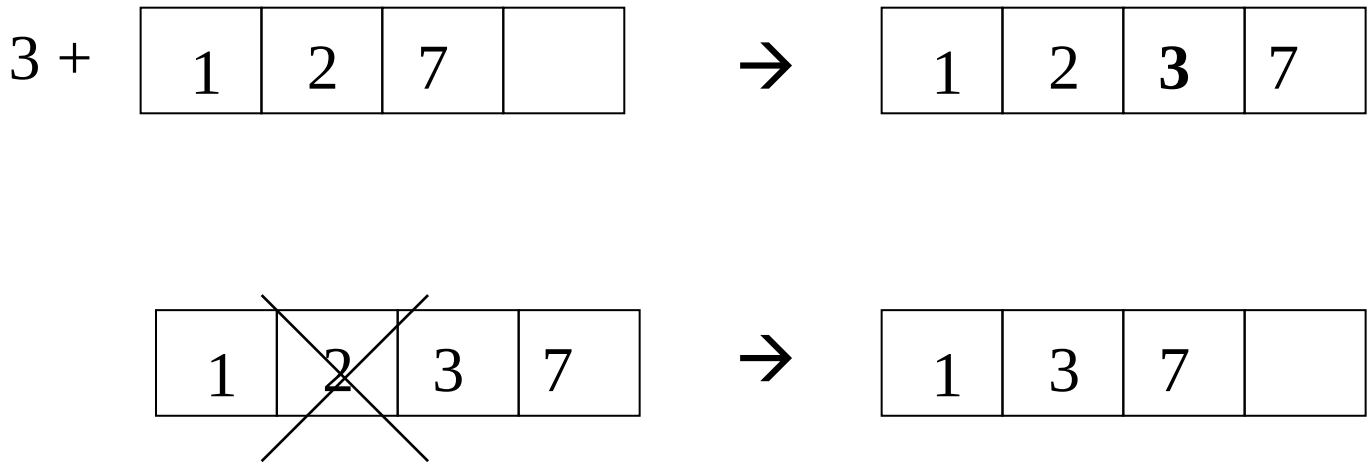
Un esempio di B+Tree di ordine 5



Che percorso compio durante la ricerca di 45?

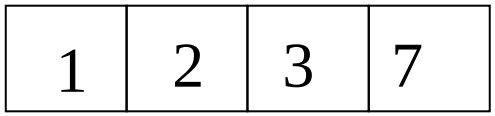
Inserimento e cancellazione

- L’inserimento o la cancellazione di una key spesso interessano solo le foglie:

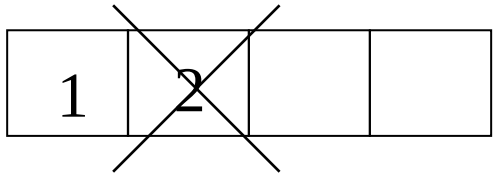


Inserimento e cancellazione 2

- Pero’ capita che l’inserimento o la cancellazione di key richiedano delle modifiche anche ai nodi interni.

9 + 

→ ?



→ Half full requirement violato

Vediamo piu’ in dettaglio gli algoritmi di inserimento e cancellazione: →

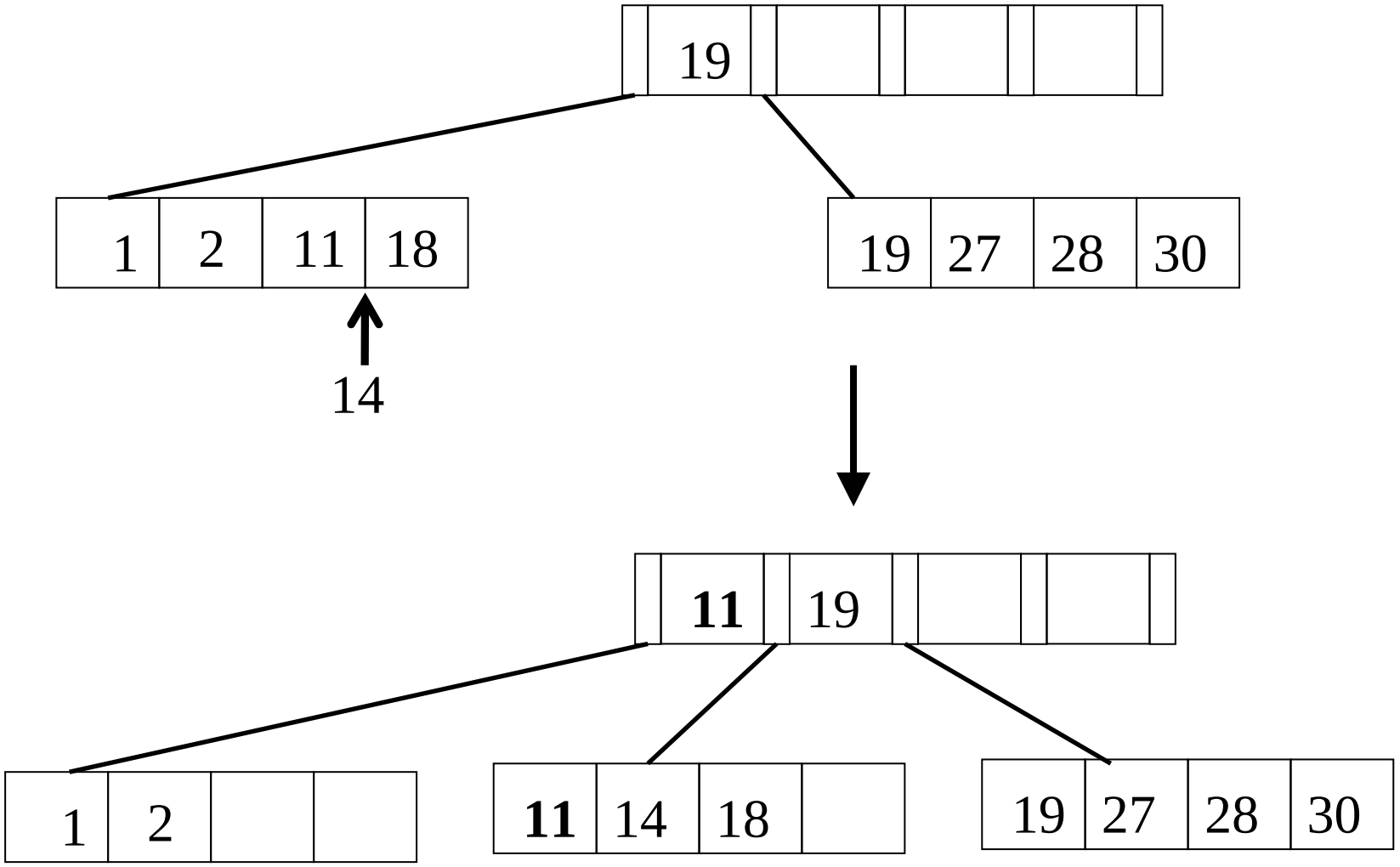
Algoritmo di inserimento

- Si ricerca la foglia che dovrebbe contenere la key da inserire
- Se nella foglia c’e’ spazio, viene inserita la key
- Altrimenti: bisogna effettuare uno split (il nodo viene diviso in due)

N.B.: Lo SPLITTING e' UGUALE ai
B tree
per i nodi INTERMEDI

ma è DIVERSO PER LE FOGLIE
(valore che viene promosso resta anche nella foglia)

Split di una foglia

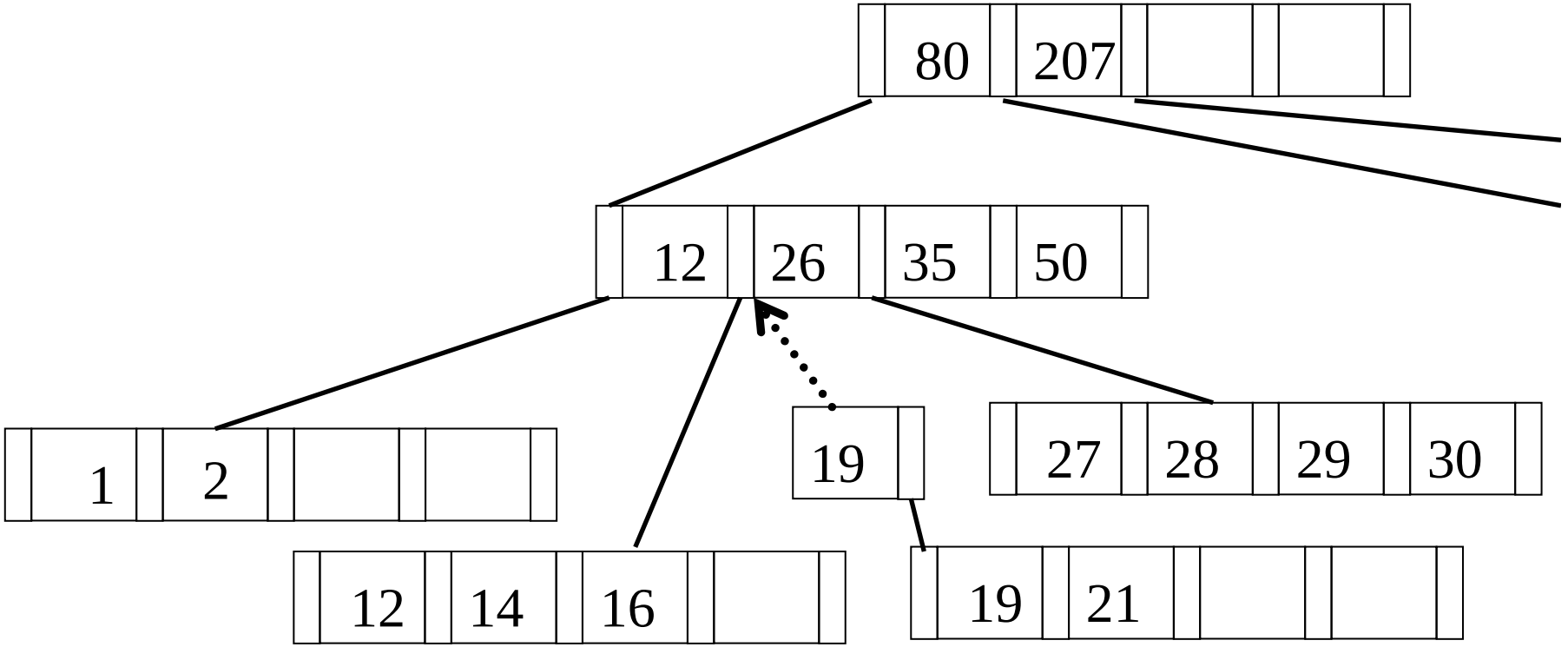


Split di una foglia 2

- La key piu’ piccola (11) della seconda nuova foglia (quella che contiene le key piu’ elevate) viene inserita nel nodo padre
- Come sono ripartiti le key del vecchio nodo nei nuovi? Perche 11 e’ andato a dx e non a sx?
  - Le key vengono suddivise a meta’, se sono dispari si assegna o a dx o a sx una key in piu’. Dove? La scelta e’ libera, purche’ si rimanga coerenti all’interno dello stesso albero
- Se il nodo padre fosse stato pieno, dove avrei inserito la key 11? →

Split di un nodo interno

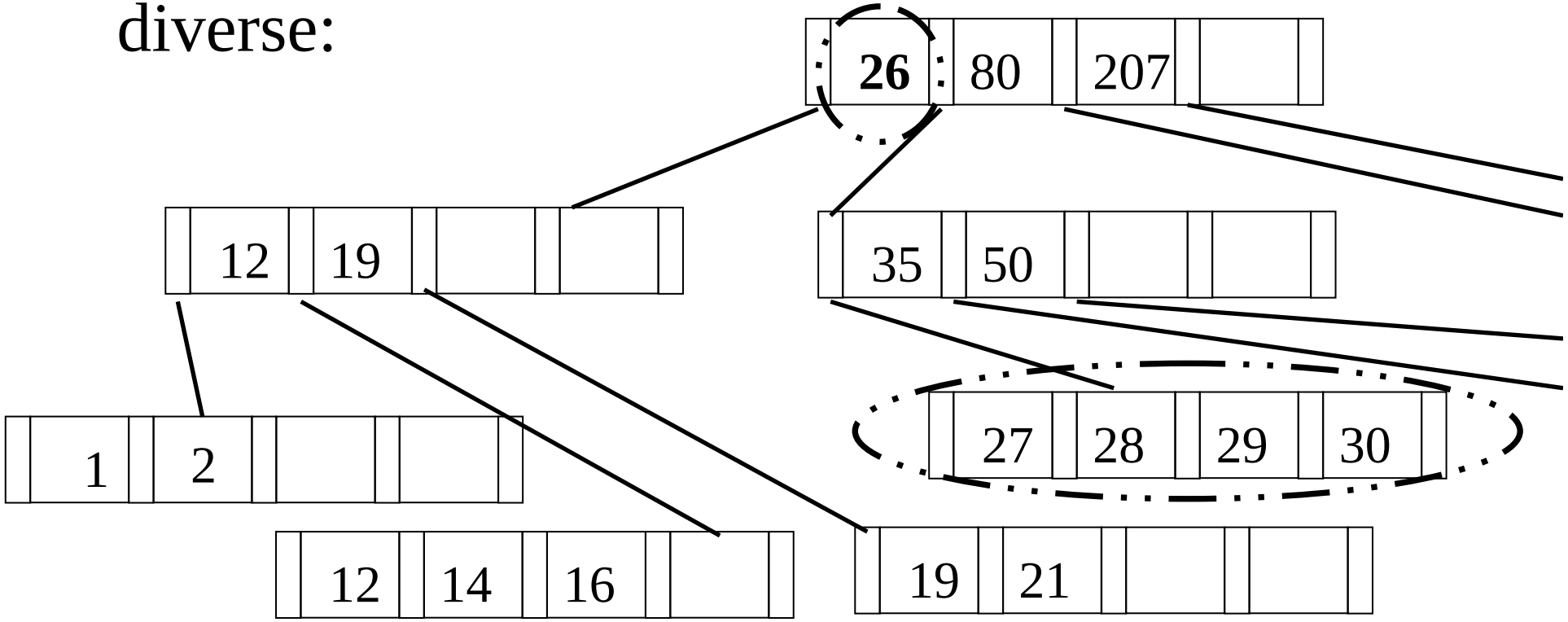
- In un nodo interno pieno si richiede di inserire una chiave:



- Bisogna dividere il nodo interno

Split di un nodo interno 2

- Lo split per un nodo interno pero’ segue regole diverse:



- Notate lo spostamento della chiave **26** e del nodo figlio ad esso associato:

→

Split di un nodo interno 3

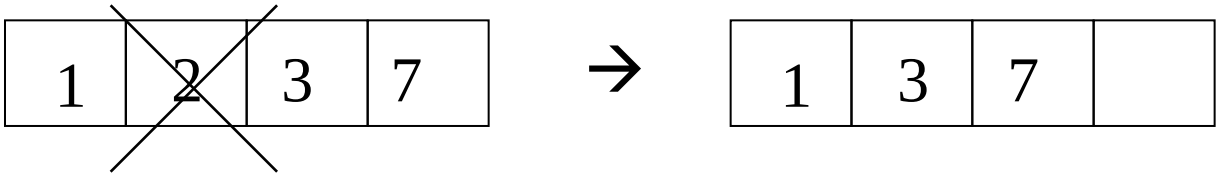
- Passi:
  - Il nodo interno viene diviso in due parti
  - Una delle chiavi centrali viene inserita nel nodo padre (nell'esempio la chiave 26)
  - Il nodo figlio, il cui puntatore era associato alla chiave spostata in alto (quello che nell'esempio inizia con la key 27), diventa il primo figlio del nuovo nodo di dx
- In questo maniera permangono le proprieta' di ordinamento che permettono di effettuare le ricerche

Split di un nodo interno 4

- Alcune annotazioni:
  - Lo split di un nodo figlio causa l’inserimento di una key nel nodo padre. Se il nodo padre e’ pieno, dovra’ a sua volta dovra’ subire uno split
  - Se l’oggetto dello split e’ il nodo radice, la key che normalmente viene inserita nel nodo padre, in questo caso viene inserita in un nodo creato apposta, il quale diventa la nuova radice. Le due “meta’ ” della vecchia radice diventano i figli della nuova radice.

Algoritmo di cancellazione

- Si ricerca la foglia che dovrebbe contenere la key da cancellare
- La key (se presente) viene cancellata



- Se la nuova situazione non viola l'half full requirement ci si ferma, altrimenti … →

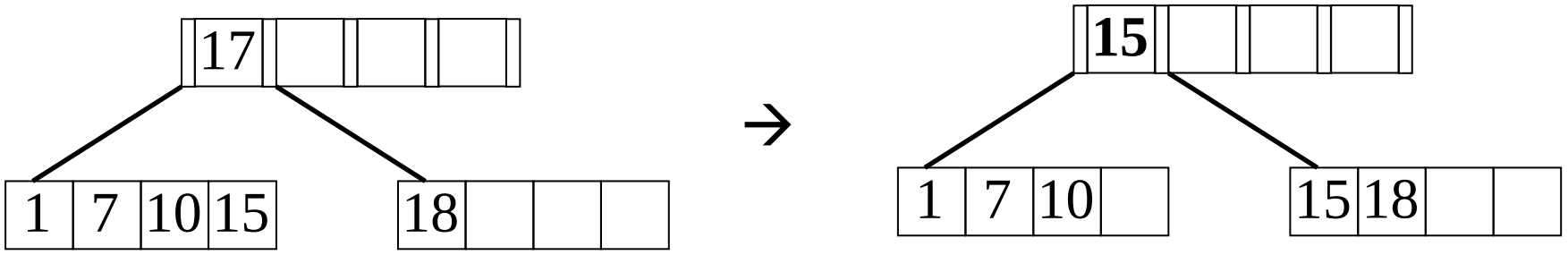
Half Full Requirement

- Ogni nodo deve essere mezzo pieno (half full requirement)
- Un nodo che per effetto delle cancellazioni si ritrova a violare l’ Half Full Requirement puo’:
  - Prendere a prestito key dai suoi vicini piu’ ricchi
  - Fondersi con un vicino povero

Key prese a prestito

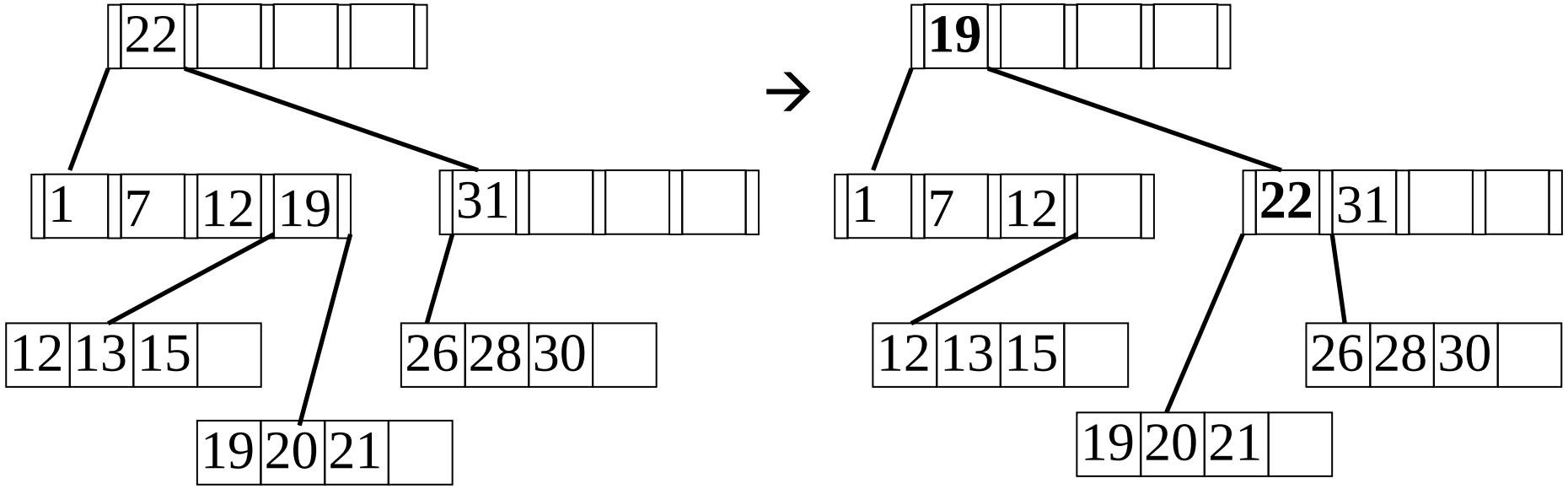
- Possono essere richieste solo ad un nodo dello stesso livello “ricco”
  - Ricco → privato della chiave non deve violare l’Half Full Requirement
  - La modalita’ varia a seconda che si tratti di prendere a prestito key da foglie o da nodi interni

Key a prestito da foglie



- Per un prestito da sx verso dx (come nell'esempio) si prende l'elemento maggiore della foglia (15) di sx
- Per un prestidto da dx verso sx si prende l'elemento minore della foglia di dx
- Bisogna aggiornare la chiave del nodo padre (per esempio ponendola uguale al piu' piccolo valore della foglia di sx
- E se le due foglie non hanno lo stesso padre? L'operazione si complica. E' necessario aggiornare le key dei nodi interni (vedi slide successive)

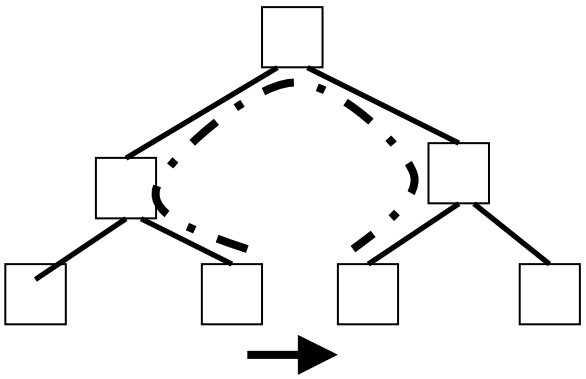
Key a prestito da nodi interni



- Alcune foglie dell’albero sono state omesse
- In grassetto le chiavi che hanno cambiato posto

Key a prestito da nodi interni 2

- Dati due nodi A,B che devono scambiarsi le chiavi, se non sono figli dello stesso padre, il riarrangiamento delle chiavi interessa tutti i nodi che congiungono A e B con il loro primo avo comune.

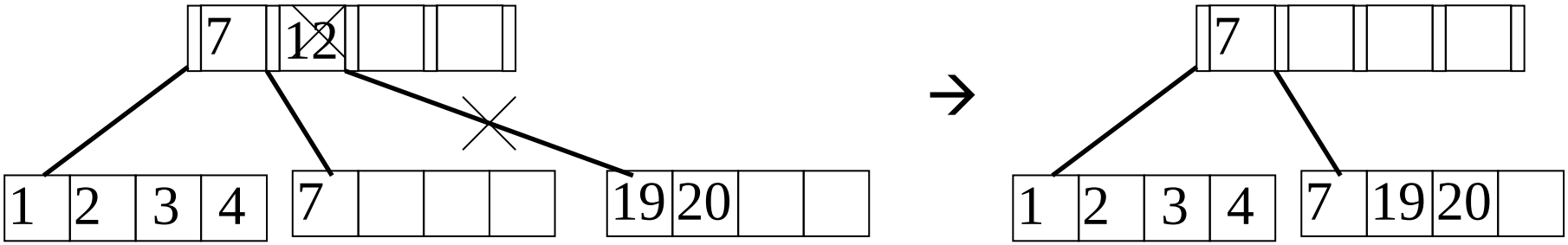


- Il procedimento applicato ad ogni nodo e' identico a quello visto nella slide precedente
- L'overhead e' talmente elevato che spesso gli algoritmi non implementato qs. scambio → si limitano i prestiti di chiavi a nodi figli dello stesso padre

Fusione di due nodi vicini

- Se un nodo viene a violare l'Half Full requirement, i suoi vicini non hanno key da prestare (perche' hanno esattamente m/2 key) → Si fondono due nodi in uno
- Il nodo risultante avra' un numero di key compreso tra m ed m/2 (un nodo aveva m/2 key, l'altro ne aveva meno di m/2)

Fusione di due nodi vicini



- Nella fusione tra due nodi vicini, la key che discriminava i due nodi ed il puntatore al secondo nodo vengono eliminati