ALBERI M-ARY

- Un albero m-ario rappresenta una generalizzazione di un albero di ricerca binario in cui ogni nodo ha al più m figli.

  Le proprietà e gli algoritmi degli alberi di ricerca binari si generalizzano banalmente.

- Se consideriamo algoritmi per memoria secondaria, tali alberi possono essere generalizzati pensando ai nodi come a blocchi di memoria.

- Un nodo interno conterrà $m - 1$ chiavi che separano i sottoalberi discendenti e m puntatori ai figli.

$\boxed{\text{ALBERI M-ARY}}$

- Anche le foglie sono blocchi che contengono gli elementi del file principale o contengono puntatori a tali elementi (se più di un elemento ha la stessa chiave).

- Un albero di ricerca binario bilanciato ha altezza $\log_2 n$, dove n è il numero di elementi.

  Un albero di ricerca m-ario bilanciato ha altezza $\log_m n$, dove n è il numero di blocchi.

- m non può essere fissato in modo arbitrario. Il valore ottimale di m in genere dipende dal sistema operativo.

B-ALBERI

- i B-alberi sono alberi m-ari bilanciati, utilizzati come strutture di indicizzazione per dati in memoria secondaria

- diretta generalizzazione degli alberi 2-3

- requisiti fondamentali per indici per memoria secondaria:

  bilanciamento: l'indice deve essere bilanciato rispetto ai blocchi e non ai singoli nodi (è il numero di blocchi acceduti a determinare il costo I/O di una ricerca)

  occupazione minima: è importante che si possa stabilire un limite inferiore all'utilizzazione dei blocchi, per evitare una sottoutilizzazione della memoria

  efficienza di aggiornamento le operazioni di aggiornamento devono avere costo limitato

B-ALBERI

- garantiscono un'occupazione di memoria almeno del 50% (almeno metà di ogni pagina allocata è effettivamente occupata)

- consentono di effettuare l'operazione di ricerca con costo, nel caso peggiore, logaritmico nella cardinalità dell'indice (cioè nel numero di valori distinti della chiave di ricerca)

- in un B-albero ogni nodo ha al più m figli, dove m è l'unico parametro dipendente dalle caratteristiche della memoria, cioè dalla dimensione del blocco

B-ALBERI

- i nodi foglia sono spesso collegati a lista, per facilitare ricerche per intervalli di chiavi o sequenziali

- parziale duplicazione delle chiavi: le entrate dell'indice (chiavi e riferimenti ai dati) sono solo nelle foglie $\Rightarrow$ la ricerca di una chiave deve individuare una foglia

- il sottoalbero sinistro di un separatore contiene valori di chiave minori del separatore, quello destro valori di chiave maggiori od uguali al separatore

B-ALBERI

Un B^+-albero di ordine m ($m \geq 3$) è un albero bilanciato che soddisfa le seguenti proprietà:

- ogni nodo contiene al più $m - 1$ elementi

- ogni nodo, tranne la radice, contiene almeno $\lceil m/2 \rceil - 1$ elementi, la radice può contenere anche un solo elemento (quindi, solo due figli)

- ogni nodo non foglia contenente j elementi ha $j + 1$ figli

- ogni nodo foglia ha una struttura del tipo:

  $$(k_1, r_1)(k_2, r_2) \ldots (k_j, r_j)$$

  dove j è il numero degli elementi del nodo (il numero massimo b_{max} di elementi contenuti in un nodo foglia può essere diverso da $m - 1$)

B-ALBERI

- nel nodo foglia $(k_1, r_1)(k_2, r_2) \ldots (k_j, r_j)$
  - $k_1, \ldots, k_j$ sono chiavi ordinate, cioè $k_1 < k_2 < \ldots < k_j$
  - r_i, $i = 1, ..., j$ rappresenta i dati satellite dell'elemento con chiave k_i, supponendo che tutti gli elementi abbiamo chiavi distinte (le foglie corrispondono al file dati)

- ogni nodo foglia ha un puntatore al nodo foglia precedente e successivo

- ogni nodo non foglia ha una struttura del tipo:

$$p_0 k_1 p_1 k_2 p_2 \cdots p_{j-1} k_j p_j$$

  dove j è il numero degli elementi del nodo

B-ALBERI

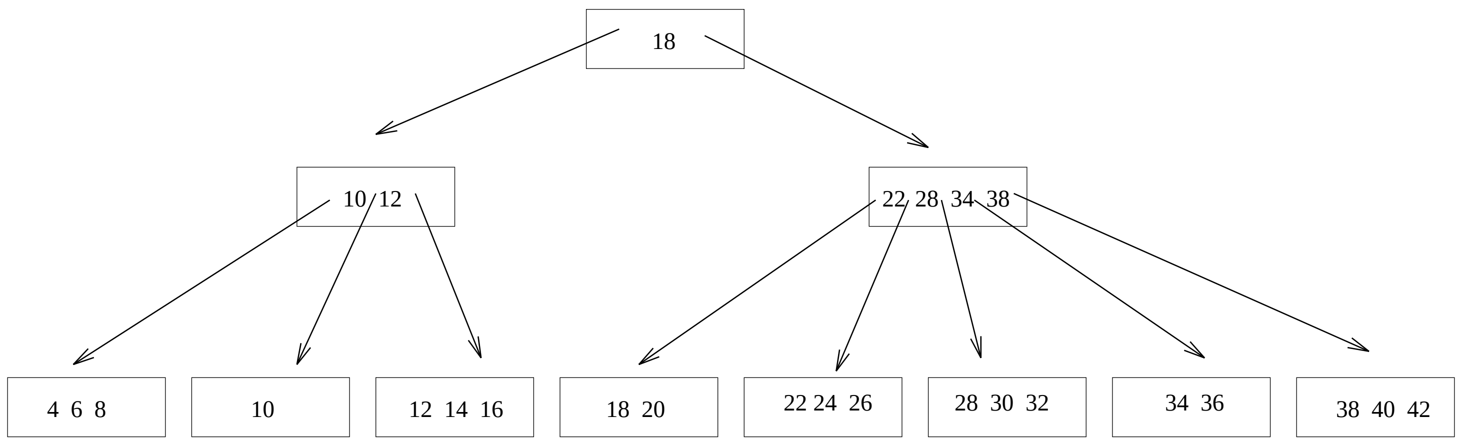
- nel nodo non foglia

$$p_0 k_1 p_1 k_2 p_2 \dots p_{j-1} k_j p_j$$

  - $k_1, \dots, k_j$ sono chiavi ordinate, cioè $k_1 < k_2 < \dots < k_j$
  - nel nodo sono presenti $j+1$ riferimenti ai nodi figli $p_0, \dots, p_j$
  - per ogni nodo non foglia, detto $K(p_i)$ $(i = 0, \dots, j)$ l'insieme delle chiavi memorizzate nel sottoalbero di radice p_i, si ha:
    * $\forall y \in K(p_0), y < k_1$
    * $\forall y \in K(p_i), k_i \leq y < k_{i+1}, i = 1, \dots, j-1$
    * $\forall y \in K(p_j), y \geq k_j$
  - ogni k_i, per $i = 1, \dots, j$ è l'elemento minimo di $K(p_i)$

B-ALBERI

Esempio di B$^+$-albero di ordine 5 con $b_{max} = 3$.



$\boxed{\text{B-ALBERI}}$

Altezza

- altezza = numero di nodi che compaiono in un cammino dalla radice ad un nodo foglia =h

- Se ogni nodo foglia contiene in media b valori, allora il B-albero avrà $\lceil n/b \rceil$ foglie.

  Se assumiamo che ogni nodo abbia il minor numero di figli possibile

  $\Rightarrow$ il livello immediatamente superiore alle foglie contiene al più $\lceil n/b \rceil/(m/2)$ nodi.

  il livello ancora precedente $\lceil n/b \rceil/(m/2)^2$.

  Allora alla radice vale $\lceil n/b \rceil/(m/2)^{h-1} \geq 1$

  Quindi $\lceil n/b \rceil \geq (m/2)^{h-1}$ e quindi $h \leq 1 + \log_{m/2}\lceil n/b \rceil$.

$\boxed{\text{B-ALBERI}}$

RICERCA

- ricerca di un elemento x: si segue il cammino dalla radice alla foglia che contiene x, se esiste:
  - viene trasferita la radice in memoria. Tre casi:
    1. se $k_i \leq x \leq k_{i+1}$, si prosegue nel nodo puntato da p_i e si ripete il processo
    2. se $x < k_1$ si prosegue nel nodo puntato da p_0
    3. se $x \geq k_n$ si prosegue nel nodo puntato da p_n
  - il costo della ricerca di una chiave è il numero di nodi letti, cioè $C_{ricerca} = h$

$\boxed{\text{B-ALBERI}}$

INSERIMENTO

- idea chiave di inserimento e cancellazione: le modifiche partono sempre dalle foglie e l'albero cresce o si accorcia verso l'alto

- esempio: inserimento

  – non si creano nuovi figli dalle foglie, ma, se necessario, si crea una nuova foglia allo stesso livello delle altre e si propaga un valore di chiave (separatore) verso l'alto

  – i nodi ai livelli superiori non sono necessariamente pieni e quindi possono "assorbire" le informazioni che si propagano dalle foglie

  – la propagazione degli effetti sino alla radice può provocare l'aumento dell'altezza dell'albero

B-ALBERI

INSERIMENTO

- l'inserimento richiede prima di tutto un'operazione di ricerca per verificare se l'elemento è già presente nell'albero

- l'inserimento avviene quindi sempre in una foglia - ci possono essere due casi:

  1. se la foglia non è piena, si inserisce la chiave e si riscrive la foglia così aggiornata (nessuna modifica ai nodi antenati)

  2. se la foglia è piena, si attiva un processo di *suddivisione* (splitting) che può propagarsi al livello superiore e, nel caso peggiore, propagarsi fino alla radice

B-ALBERI

INSERIMENTO: suddivisione

- P un nodo pieno in cui deve essere inserita una chiave

- sequenza ordinata di m entrate che si verrebbero a creare

  $$(p_0)k_1p_1k_2p_2 \ldots k_gp_gk_{g+1}p_{g+1}k_{g+2}p_{g+2} \ldots k_hp_h$$

  con $h = m - 1$ se il nodo non è foglia, $h = b_{max}$ altrimenti e $g = \lceil m/2 \rceil - 1$ se il nodo non è foglia $g = \lceil b_{max}/2 \rceil$ se il nodo è foglia

- se il nodo è foglia, si partizionano le chiavi come segue:

  – nel nodo P gli elementi:

  $k_1p_1k_2p_2 \ldots k_gp_g$

– in un nuovo nodo P' gli elementi:

$k_{g+1} p_{g+1} k_{g+2} p_{g+2} \dots k_h p_h$

- se il nodo non è foglia, si partizionano le chiavi come segue:

  – nel nodo P gli elementi:

  $p_0 k_1 p_1 k_2 p_2 \dots k_g p_g$

  – in un nuovo nodo P' gli elementi:

  $p_{g+1} k_{g+2} p_{g+2} \dots k_h p_h$

B-ALBERI

INSERIMENTO: suddivisione

- nel nodo Q padre di P si inserisce l'entrata $k_{g+1}p'$ con p' puntatore al nodo P'

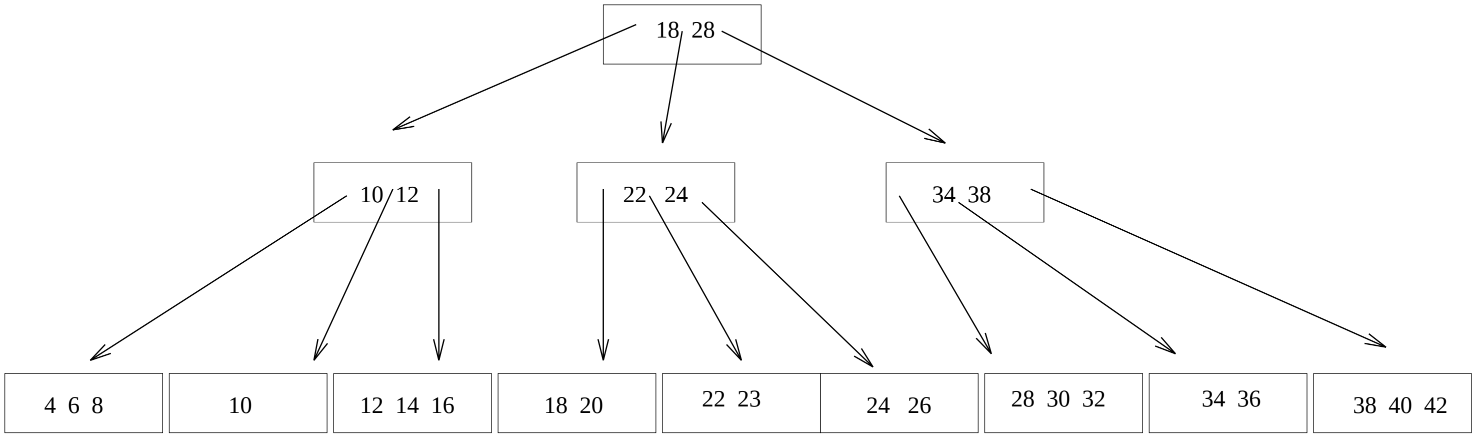
- se anche Q è pieno, allora il processo di splitting deve essere ripetuto

- se si deve sdoppiare la radice, questa diventa $pk_{g+1}p'$ con p puntatore al nodo P (la radice prima dell'inserzione)

  unico caso di nodo con meno di $\lceil m/2 \rceil$ figli

B-ALBERI

INSERIMENTO (di 23) nell'albero iniziale



costi:

- caso migliore (assenza di splitting): si leggono h nodi e si riscrive una foglia, quindi $C_{inserimento}^{min} = h + 1$

- caso peggiore (lo splitting si propaga fino alla radice): si leggono h nodi e se ne riscrivono $2h + 1$, quindi $C_{inserimento}^{max} = 3h + 1$

$\boxed{\text{B-ALBERI}}$

CANCELLAZIONE

- anche in questo caso ci si riconduce ad una cancellazione da un nodo foglia

- viene innanzitutto effettuata la ricerca dell'elemento da cancellare nell'albero

$\boxed{\text{B-ALBERI}}$
CANCELLAZIONE

- Sia L il nodo foglia che contiene l'elemento da cancellare:

  1. se la foglia non diventa vuota, si cancella la chiave e si riscrive la foglia così aggiornata

     – se la chiave cancellata è k_1 si devono modificare gli antenati di L. Sia P il padre di L

       * se L è puntato da p_0, allora modifico il minimo antenato di L per cui L non è il discendente più a sinistra

       * se L non è puntato da p_0, allora modifico opportunamente P.

  2. se la foglia diventa vuota, la si disalloca e si aggiustano i puntatori in P. Se P ha meno di $\lceil m/2 \rceil - 1$ elementi, si attiva un processo di *concatenazione* o un processo di *bilanciamento*

B-ALBERI

CANCELLAZIONE - concatenazione

- la concatenazione di due nodi adiacenti P e P' è possibile se i due nodi contengono, complessivamente, al più $m-1$ chiavi

- un nodo con meno di $\lceil m/2 \rceil - 1$ elementi, detto in *underflow*, si combina quindi con un nodo fratello adiacente con al più $\lfloor m/2 \rfloor + 1 = \lceil m/2 \rceil$ chiavi (per esercizio, dimostrare che la somma soddisfa le proprietà dei B-alberi)

- la concatenazione opera in maniera esattamente inversa al processo di suddivisione

B-ALBERI

CANCELLAZIONE - concatenazione

- situazione iniziale:
  - nodo P in underflow con elementi:
    $p_0 k_1 p_1 k_2 p_2 \ldots k_e p_e$ $(e = \lceil m/2 \rceil - 2)$
  - nodo P' adiacente a destra di P con elementi:
    $p'_0 k_{e+1} p_{e+1} \ldots$
  - nodo Q, padre di P e P', con elementi
    $\ldots k_{t-1} p_{t-1} k_t p_t k_{t+1} p_{t+1} \ldots$
    con p_{t-1} puntatore a P e p_t puntatore a P'

- la concatenazione dei due fratelli porta alla seguente situazione:
  - nodo P con elementi:
    $p_0 k_1 p_1 k_2 p_2 \ldots k_e p_e k_t p'_0 k_{e+1} p_{e+1} \ldots$;
  - nodo Q con elementi:
    $\ldots k_{t-1} p_{t-1} k_{t+1} p_{t+1} \ldots$
    con p_{t-1} puntatore a P

$\boxed{\text{B-ALBERI}}$

CANCELLAZIONE - concatenazione

- l'eliminazione della chiave k_t dal nodo padre può innescare a sua volta una concatenazione (o un bilanciamento)

- la concatenazione si può propagare ricorsivamente fino alla radice, causandone l'eliminazione, con conseguente diminuzione dell'altezza dell'albero

$\boxed{\text{B-ALBERI}}$

CANCELLAZIONE - **bilanciamento**

- se fra due fratelli adiacenti non si può applicare la concatenazione, allora si distribuiscono tra di essi gli elementi in modo bilanciato

- il bilanciamento interessa anche il nodo padre poiché uno dei suoi elementi viene modificato, ma il numero dei suoi elementi non cambia (si possono però dovere cambiare i valori nei nodi)

B-ALBERI

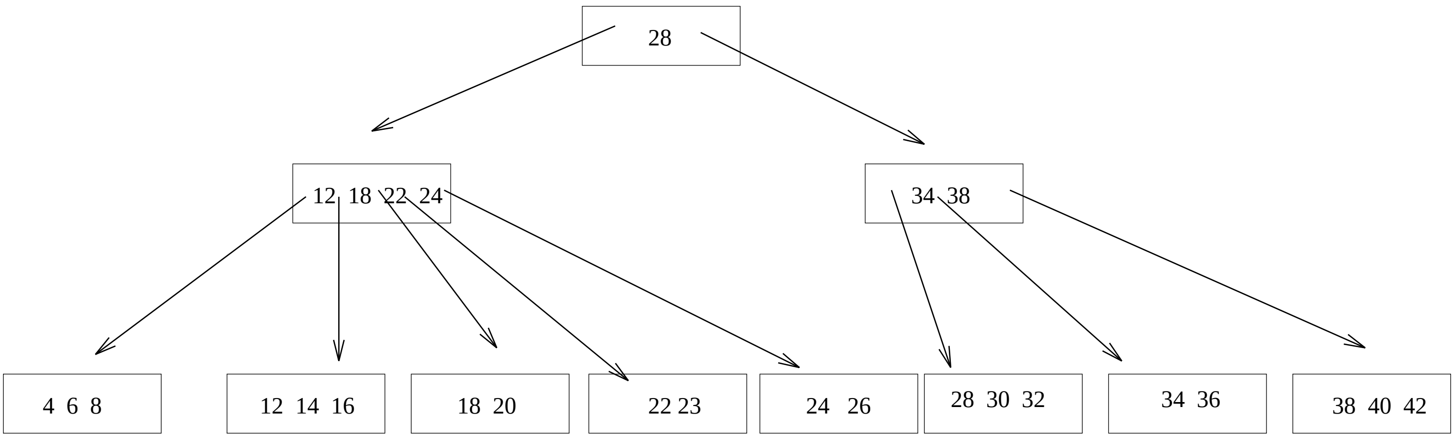
CANCELLAZIONE - bilanciamento

- situazione iniziale:
  - nodo P' in underflow con elementi:
    $p'_0 k'_1 p'_1 k'_2 p'_2 \ldots k'_j p'_j$ ($j = \lceil m/2 \rceil - 2$)
  - nodo P adiacente a sinistra a P' con elementi:
    $p_0 k_1 p_1 k_2 p_2 \ldots k_e p_e$
  - nodo Q, padre di P e P', con elementi:
    $\ldots k_{t-1} p_{t-1} k_t p_t k_{t+1} p_{t+1} \ldots$
    con p_{t-1} puntatore a P e p_t puntatore a P'

- per bilanciare gli elementi nei due nodi:
  - si considera la lista di chiavi
    $k_1\ k_2\ \ldots\ k_e\ k_t\ k'_1\ k'_2\ \ldots\ k'_j$
  - le prime $\lfloor (e+j)/2 \rfloor$ chiavi rimangono nel nodo P
  - si sostituisce nel nodo padre la chiave k_t con la chiave in posizione $\lfloor (e+j)/2 \rfloor + 1$
  - le rimanenti $\lceil (e+j)/2 \rceil$ chiavi si mettono in P'

B-ALBERI

CANCELLAZIONE

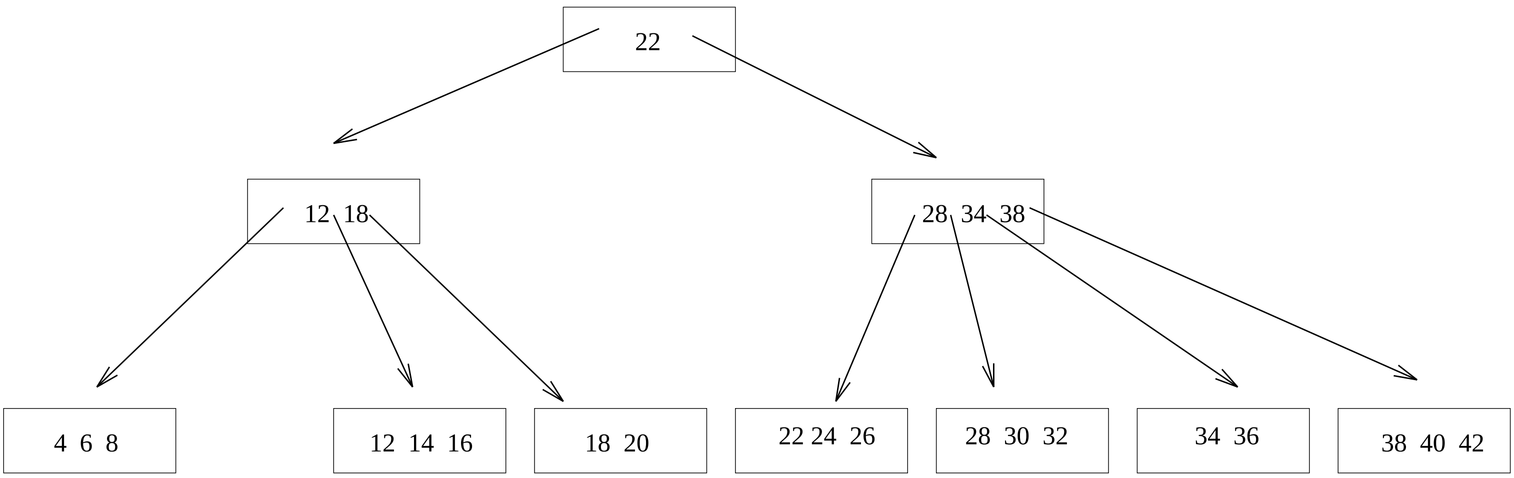


Cancellazione da un B-albero:

Concatenazione [cancellazione di 10 dall'albero dopo inserimento]

B-ALBERI

CANCELLAZIONE



Cancellazione da un B-albero:

Bilanciamento [cancellazione di 10 dall'albero di partenza]

$\boxed{\text{B-ALBERI}}$

CANCELLAZIONE - costi

- caso migliore (la cancellazione avviene in una foglia e non si rendono necessari né concatenazione né bilanciamento): si leggono h nodi e si riscrive una foglia, quindi $C^{min}_{cancellazione} = h + 1$

- caso peggiore (tutte le pagine nel percorso di ricerca devono essere concatenate ad eccezione delle prime due, il figlio della radice coinvolto nel percorso viene modificato e la radice viene quindi modificata): si leggono $2h - 1$ nodi e se ne riscrivono $h + 1$, quindi $C^{max}_{cancellazione} = 3h$

B-ALBERI

Se esistono più record con lo stesso valore chiave, i dati satellite possono essere sostituiti da puntatori al file dati. I nodi foglia avranno la seguente struttura:

- nel nodo foglia $(k_1, r_1)(k_2, r_2)\ldots(k_j, r_j)$
  - $k_1, \ldots, k_j$ sono chiavi ordinate, cioè $k_1 < k_2 < \ldots < k_j$
  - nel nodo sono presenti j riferimenti al file dei dati $r_1, \ldots, r_j$
- ogni nodo foglia ha un puntatore al nodo foglia precedente e successivo