Indeksi

Mehanizmi za ubrzavanje pristupa traženim podacima

Strategije pristupa podacima (*access methods*)

- Kako DBMS dolazi do traženih slogova?
  - Sekvencijalna pretraga je jedan način i podrazumeva čitanje podatka redom i , u najgorem slučaju, celog DB fajla (ili strukture u memoriji)
  - Da bi se ubrzalo pronalaženje i izbegla sekvencijalna pretraga DBMS može da koristi različite strategije:
    - Uredi fajl prema određenom kriterijumu, pa iskoristi algoritme kao što je binarna pretraga
    - Primeni heš fajl organizaciju
    - Koristi tehnike indeksiranja i kreira dodatne strukture koje mogu da obezbede direktan pristup podacima kao što su:
      - Heš tabele
      - Uređeni indeksi nad tabelama

UPOTREBA TEHNIKA INDEKSIRANJA

- Oni se mogu primeniti:
  - U radu sa tabelama, tako što će biti kreirani dodatni indeksni fajlovi (indeks tabele) nad određenim tabelama (CREATE INDEX)
  - U radu sa meta podacima – koji ne podrazumevaju samo meta podatke tabela, već i podatke o DB fajlovima, transkcijama, …
  - U radu sa skladištem podatka
  - Tokom izvršavanja upita, na primer pri sortiranju ili spajanju, kada se koriste kao dodatne privremene strukture u memoriji

Indeksi

- Indeks se defiiniše nad atributom ili skupom atributa, tzv. indeksnim poljem, koji predstavlja ključ po kojem se vrši pretraga indeksne strukture umesto celih fajlova sa podacima.
- Indeksni fajl se sastoji iz indeksnih zapisa (*index entry*) u formi

kojima su povezane vrednosti ključa/indeksnog polja sa informacijom o lokaciji slogova podataka (*data entry*) sa navedenom vrednošću ključa.
Informacija o lokaciji sloga – najčešće predstavlja vrstu reference na slog (RID) ili adresa bloka.

- Jednoj tabeli može biti pridruženo više indeksa.
- DBMS obezbeđuje da sadržaji tabela i indeksa budu sinhronizovani. Dakle, indeksi imaju sopstvene vremenske i prostorne troškove režije (overhead).

Vrste i merenje kvaliteta indeksa

- Dve osnovne vrste indeksa:
  - Uređeni. Uređeni prema vrednosti indeksnog polja.
  - Hash indeksi. Zasnovani na uniformnoj distribuciji vrednosti u kolekciju korpi (*bucket*-a). Određivanje pripadnosti korpi se vrši na osnovu vrednosti odabrane heš funkcije.

- Ne postoji univerzalni odabir vrste indeksa. Svaka vrsta može biti manje ili više dobra u zavisnosti od načina upotrebe.
  Procena performansi neke indeksne tehnike se vrši analizom:
  - Vrste pristupa/pretrage koje efikasno podržava (jedna vrednost, opseg vrednosti)
  - Vreme dobavljanja traženog/ih podatka
  - Vreme dodavanja – uključuje upis u fajl sa podacima i ažuriranje indeksa
  - Vreme brisanja - uključuje upis u fajl sa podacima i ažuriranje indeksa
  - Prostorno prekoračenje – dodatni prostor koji se rezerviše za sam indeks

Podela

- Primarni vs. Sekundarni

  Da li uređenje slogova u fajlu sa podacima prati uređenje indeksiranih polja u indeksu?

- Klasterovani vs. Neklasterovani (Grupišući/Negrupišući)

  Da li se su podaci u fajlu uređeni prema vrednosti ključa u indeksu ili ne?
  (klasterovani indeks nije isto što i klasterovana fajl oranizacija)

- Gusti vs. Retki

  Da li se u indeksu nalaze sve vrednosti indeksnog polja koje se pojavljuju i u fajlu sa podacima?

- Jednonivoiski vs. Višenivoiski

  Da li se u indeksnoj strukturi definiše ‘indeks nad indeksom’ ili ne?

Uređeni indeksi

O uređenim indeksima i uređenim fajlovima

Uređeni indeksi

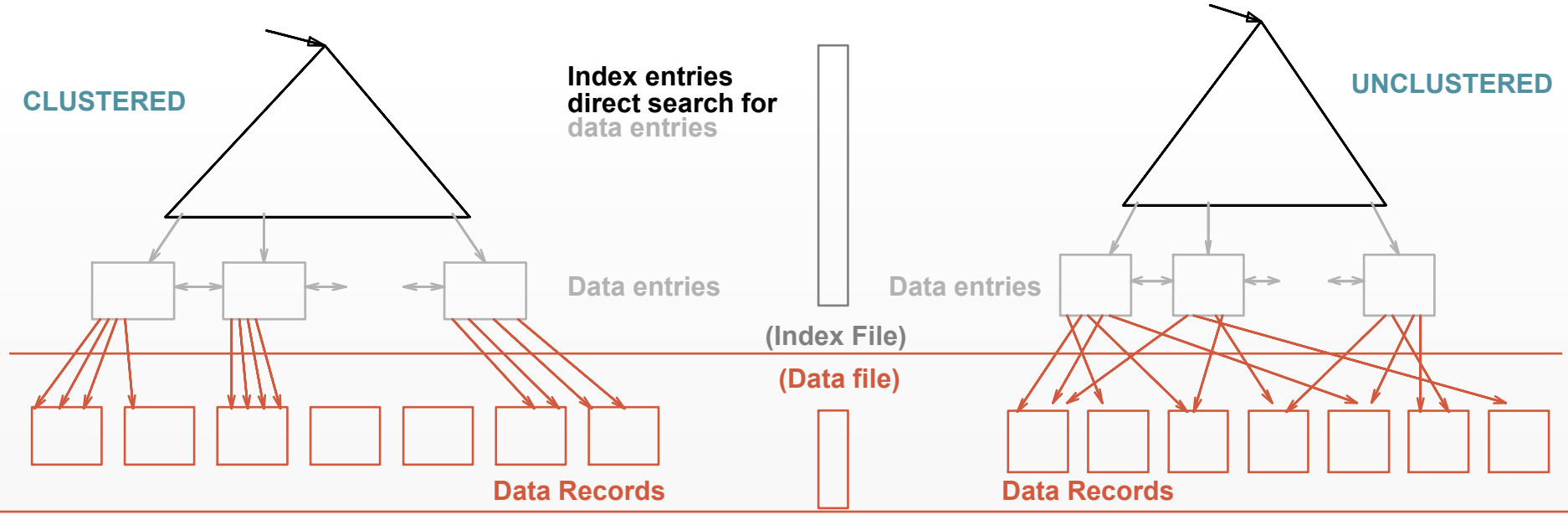
Osnovni koncept i vrste

Uređeni indeksi

- Struktura podatka u kojoj su parovi
  key, data entry
  uređeni prema vrednosti ključa.
  Key - može biti jedan ili više atributa tabele nad kojima je postavljen indeks.
  Data entry – ukazuje na torku u fajlu (tabeli).
- Struktura se beleži tako da omogući brzu pretragu.
- DBMS ima obavezu da pri izmeni sadržaja tabele sinhronizovano vrši i izmene u pridruženim indeksima.
- Tokom izvršavanja upita DBMS procenjuje koje indekse je moguće iskoristiti za ubrzavanje izvršavanja, tj. smanjenje broja čitanja/pisanja po disku.
- Ključni troškovi:
  - Memorijski prostor
  - Održavanje

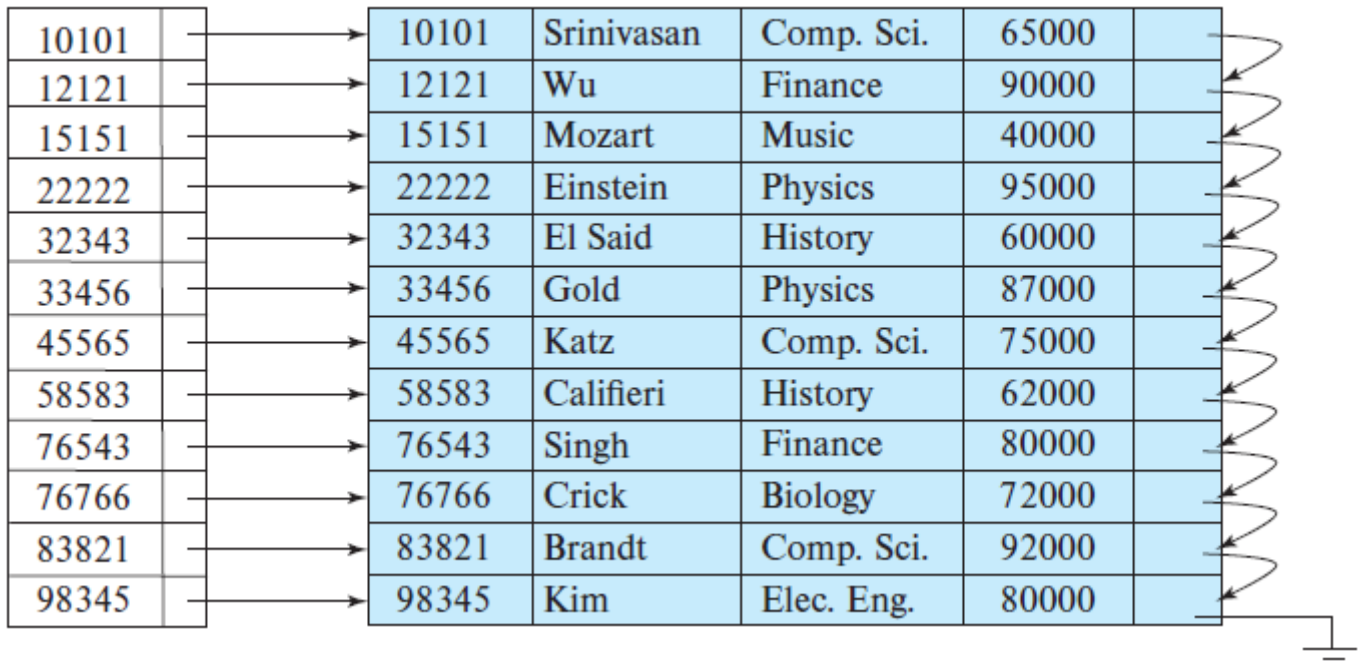
Primarni i sekundarni indeksi

- Klasterišući indeks. Indeksno polje odgovara kriterijumu po kojem je fajl sa podacima uređen. Klasterišući indeks se često naziva i primarnim. Teorijski gledano primarni indeks ne mora biti definisan nad primarnim ključem relacije, mada najčešće i jeste tako implementirano.
- Neklasterišući indeks. Indeks u kojem indeksno polje nameće uređenje koje ne odgovara sekvencijalnom uređenju fajla. Neklasterišući indeks se naziva i sekundarnim indeksom.



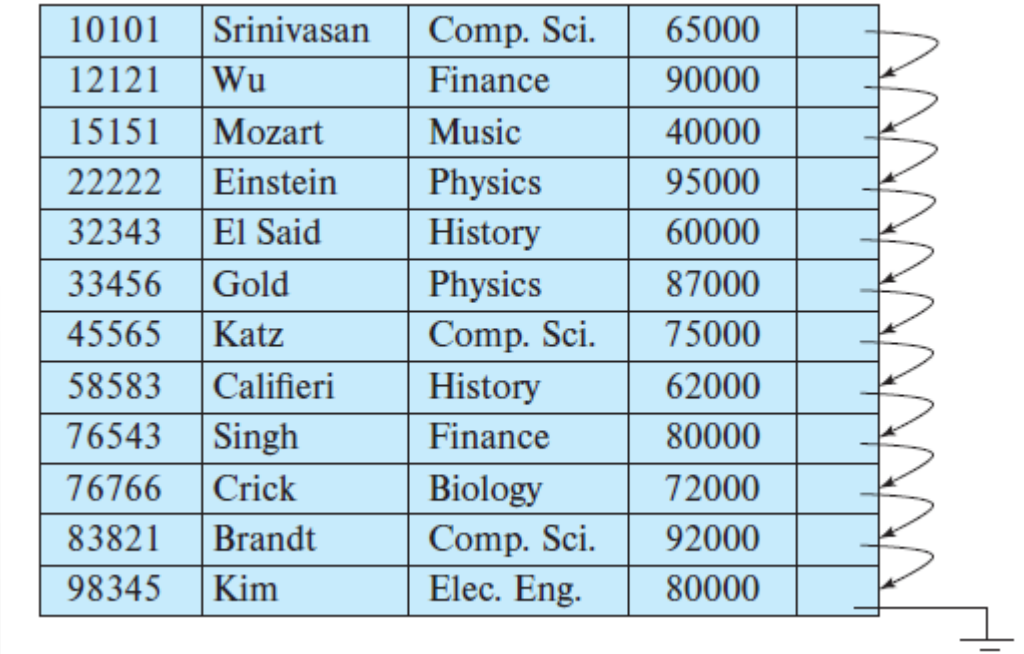
Indeks-sekvencijalni fajlovi

- Fajl sa sekvencijalnim uređenjem prema ključu nad kojim je postavljen i uređeni indeks se naziva indeks-sekvencijalnim fajlom.



10101	10101	Srinivasan	Comp. Sci.	65000
12121	12121	Wu	Finance	90000
15151	15151	Mozart	Music	40000
22222	22222	Einstein	Physics	95000
32343	32343	El Said	History	60000
33456	33456	Gold	Physics	87000
45565	45565	Katz	Comp. Sci.	75000
58583	58583	Califieri	History	62000
76543	76543	Singh	Finance	80000
76766	76766	Crick	Biology	72000
83821	83821	Brandt	Comp. Sci.	92000
98345	98345	Kim	Elec. Eng.	80000

indeks-sekvencijalni fajl



10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	

sekvencijalni fajl

Primarni indeksi

- Gusti indeks.

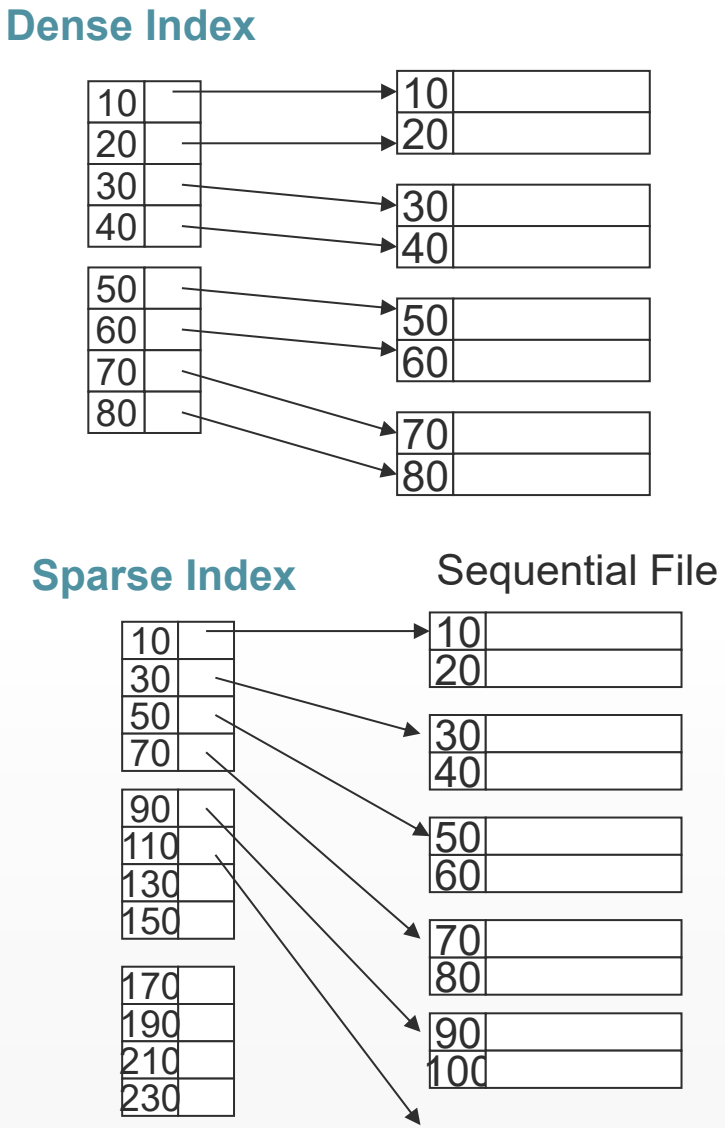
  U indeksu postoji slog za svaku vrednost indeksnog polja.

- Redak indeks.

  U indeksu postoje samo neke vrednosti indeksnog polja.

  Obično sadrže jedan ključ-pokazivač par po bloku.

  Primenljiv samo u slučaju sekvencijalnog uređenja prema indeksnom polju.



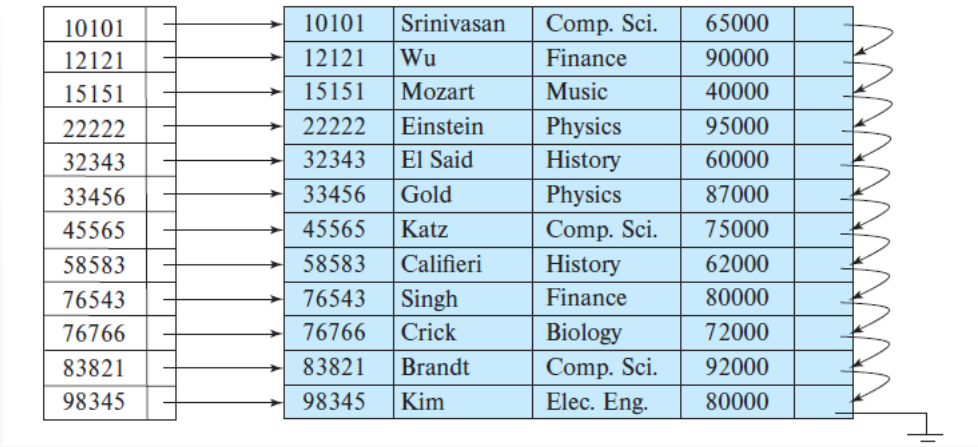
Gusti vs. Retki

- Gusti indeksi.

  Može se utvrditi da li neka vrednost indeksnog polja postoji bez učitavanja fajla sa podacima.

  Zauzima više prostora.

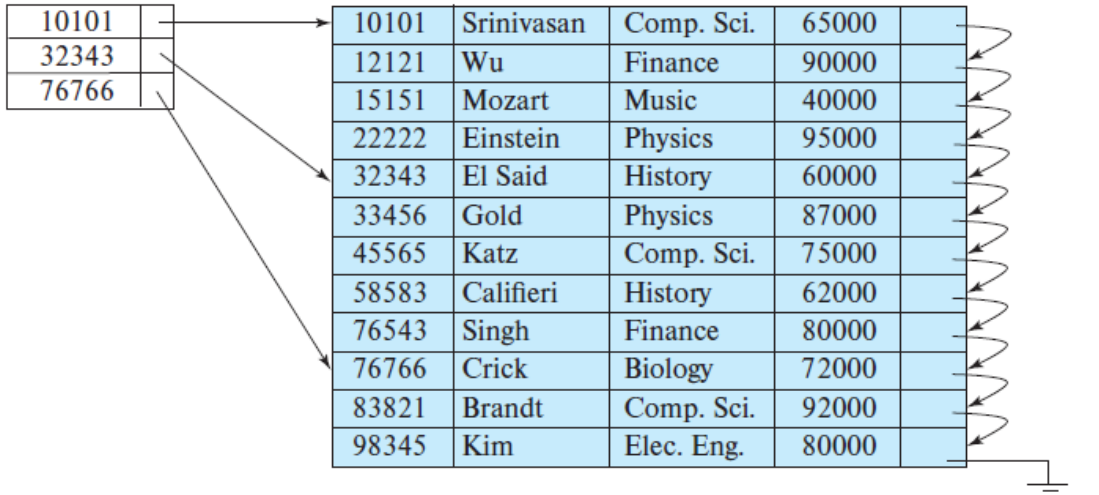
  Sporije se pretražuje sam indeks.



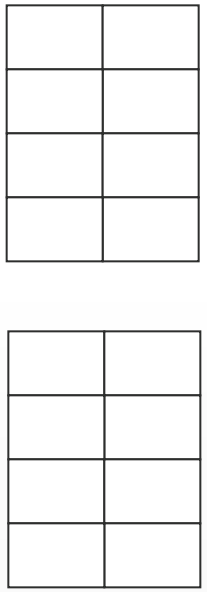
- Retki indeksi.

  Veći deo indeksa može biti učitan u memoriju.

  Lakši za implementaciju dodavanja novih slogova.



Duplikati / gust indeks



10	
10	

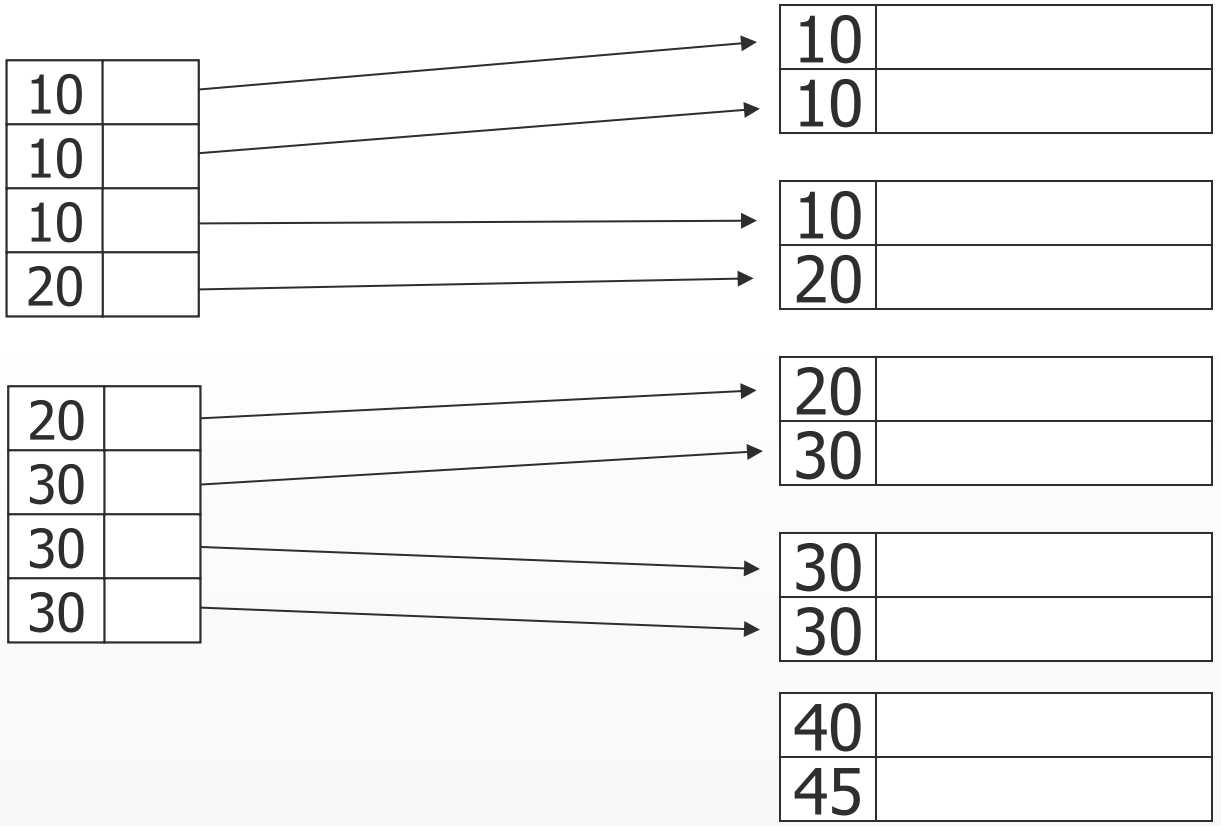
10	
20	

20	
30	

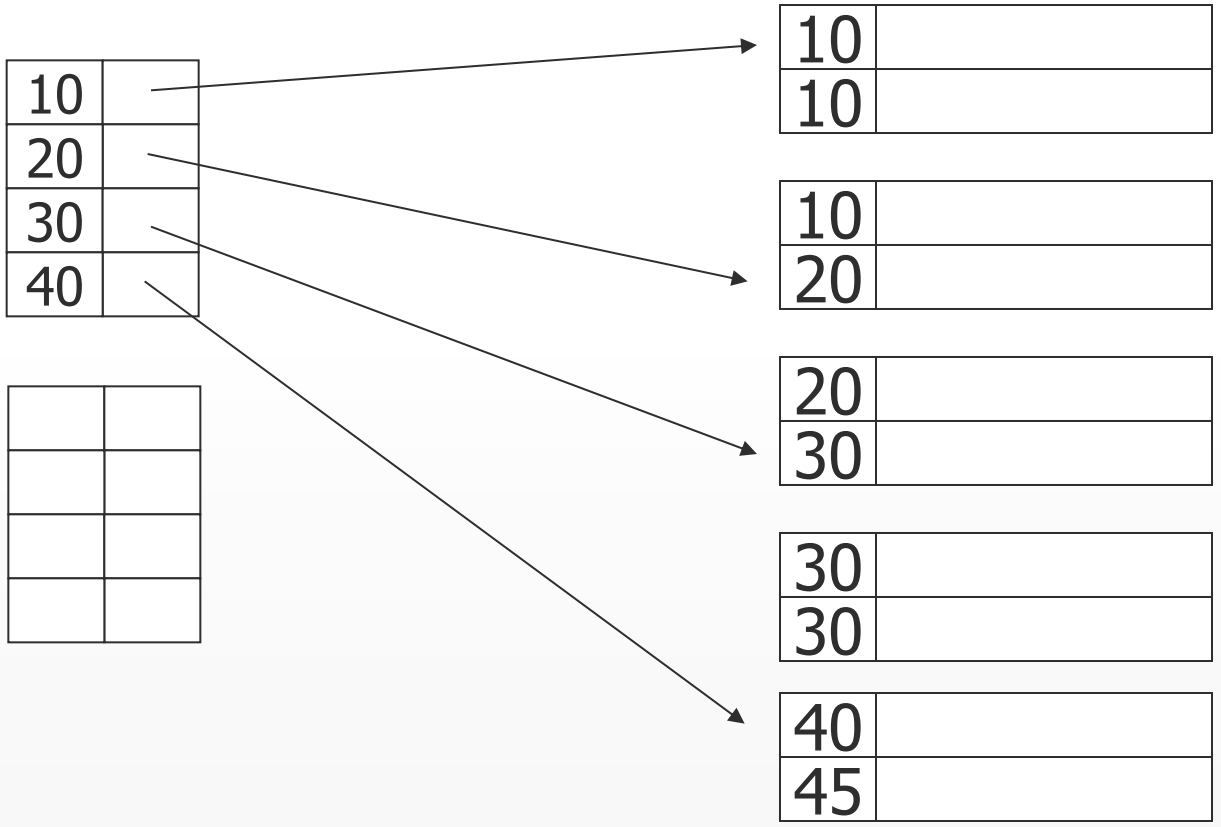
30	
30	

40	
45	

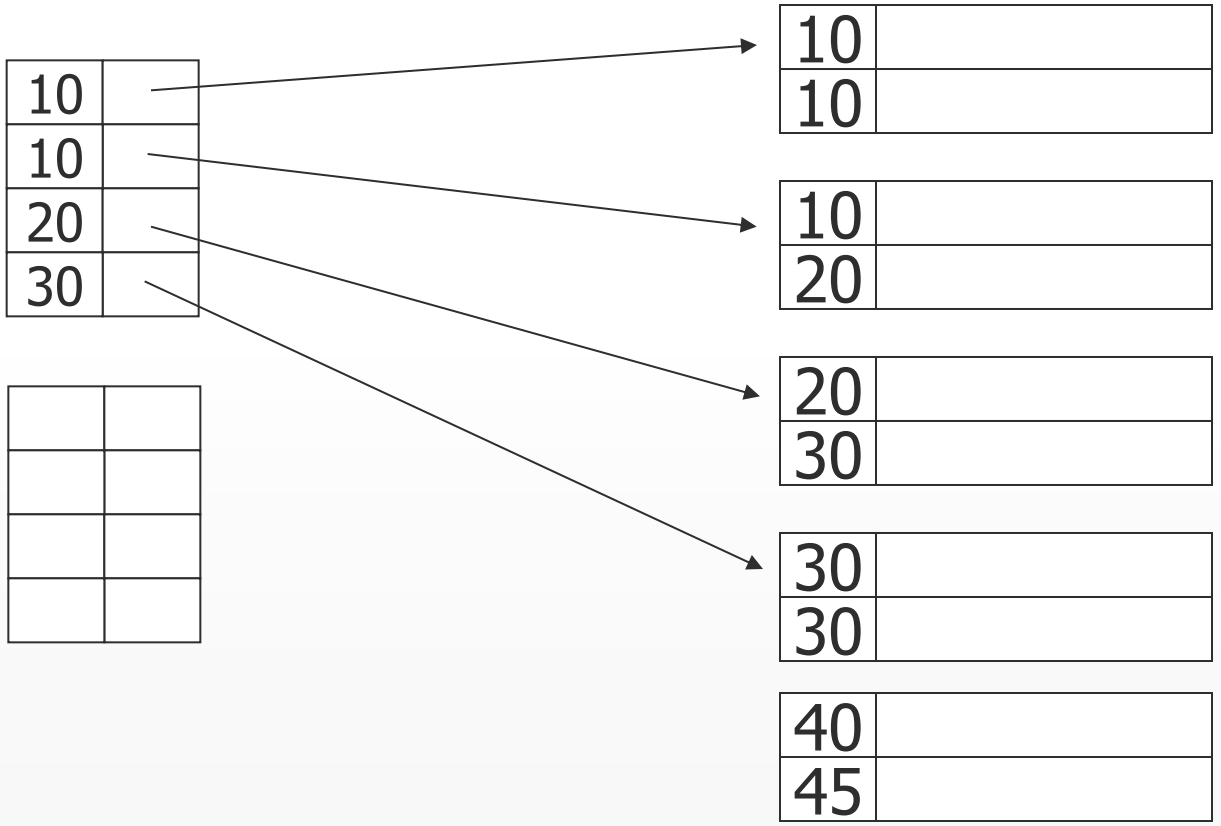
Duplikati / gust indeks – prvi način



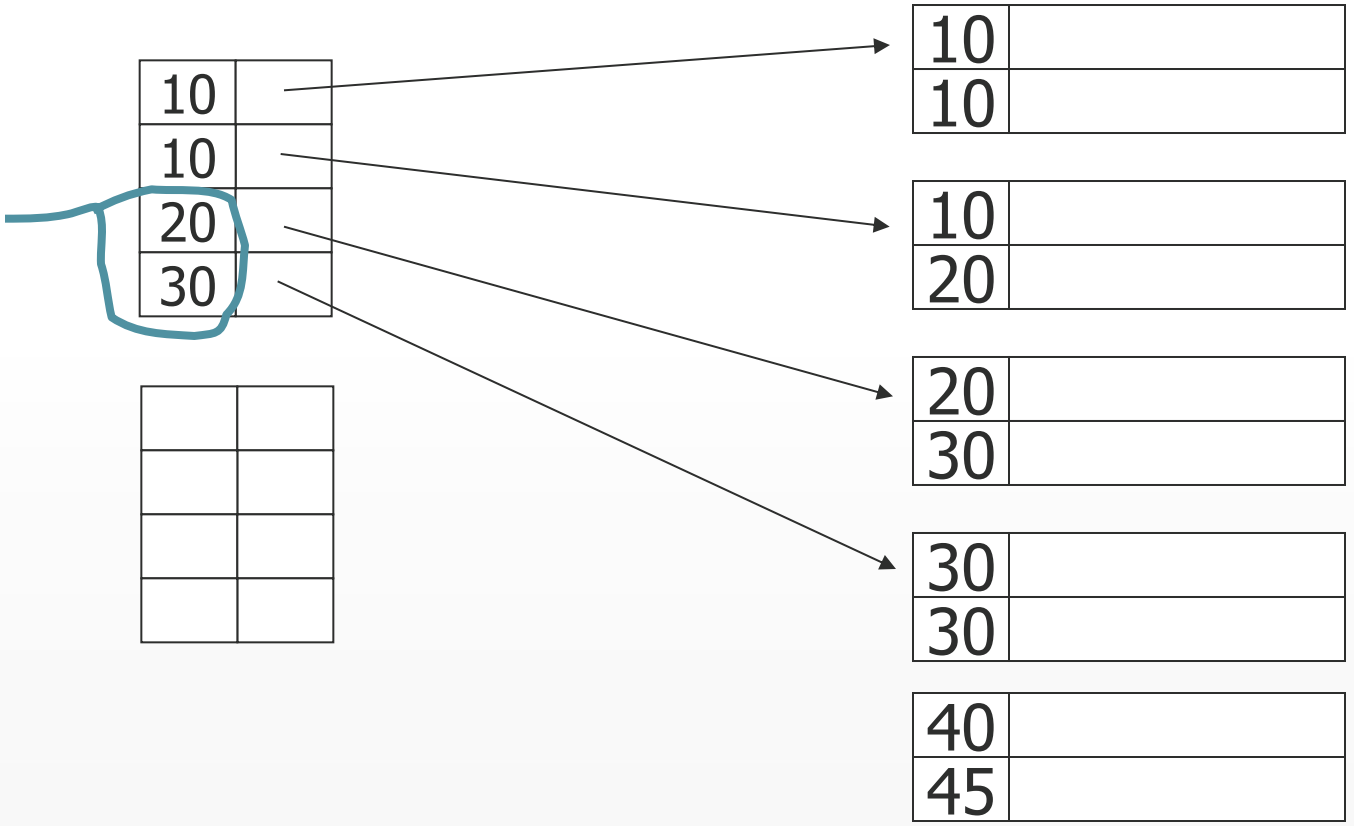
Duplikati / gust indeks – drugi način



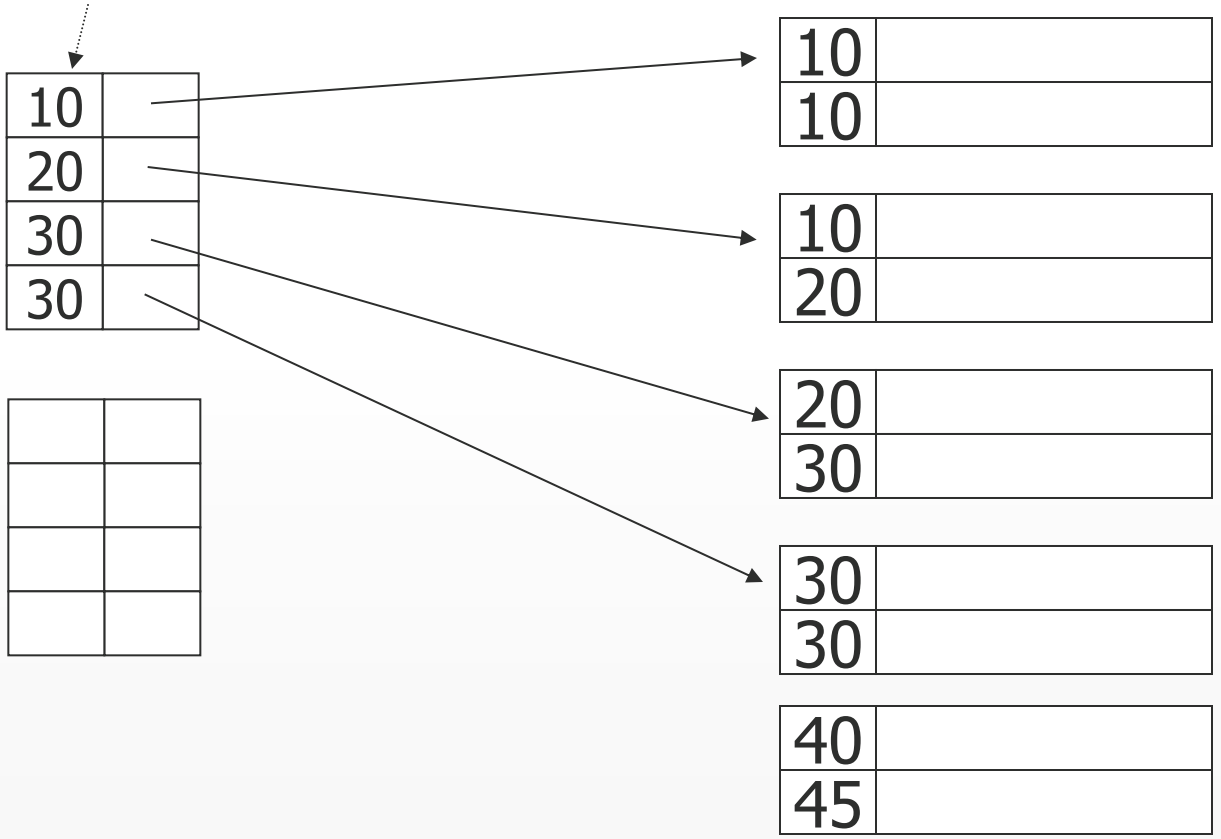
Duplikati / redak



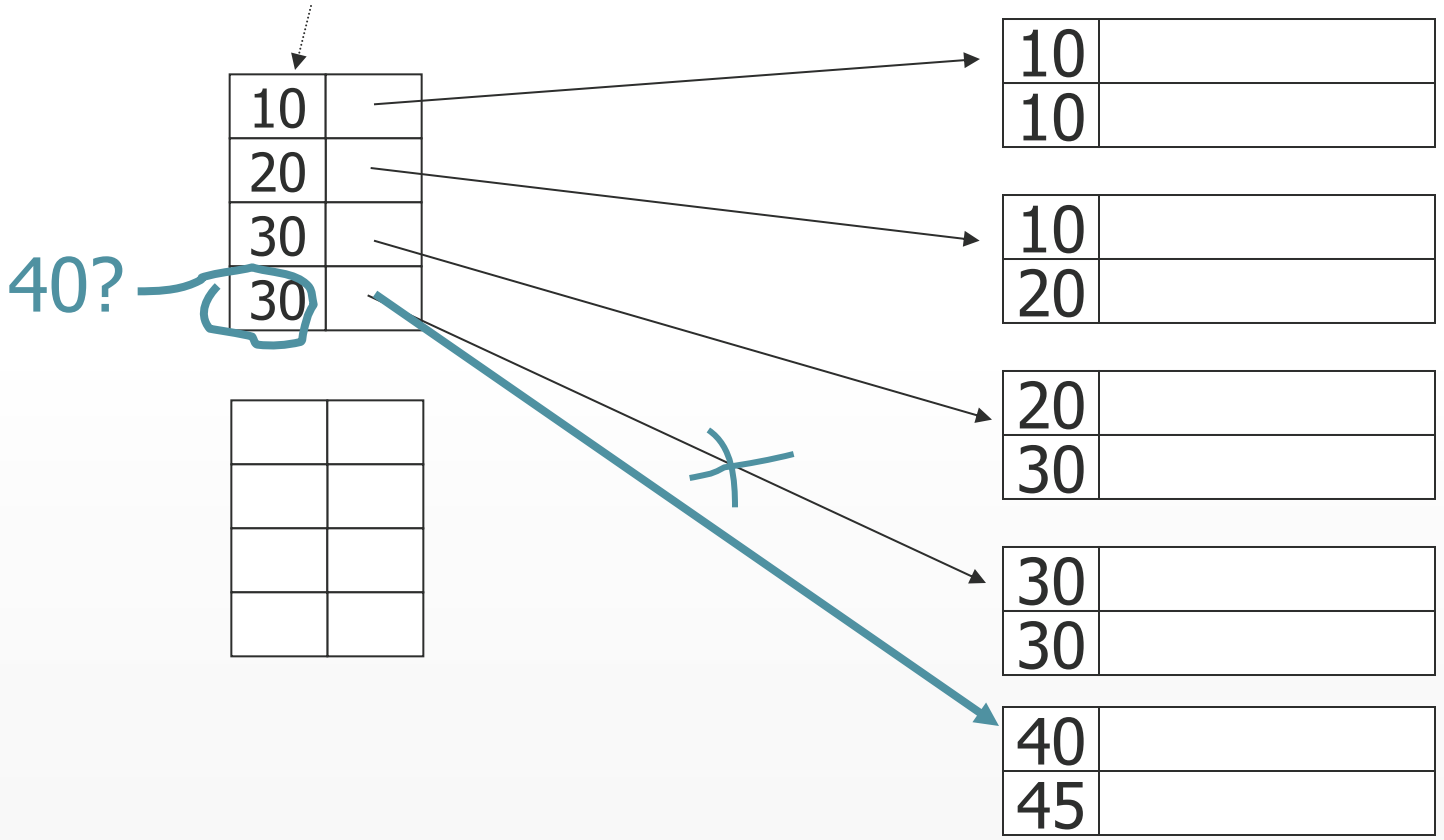
Duplikati / redak



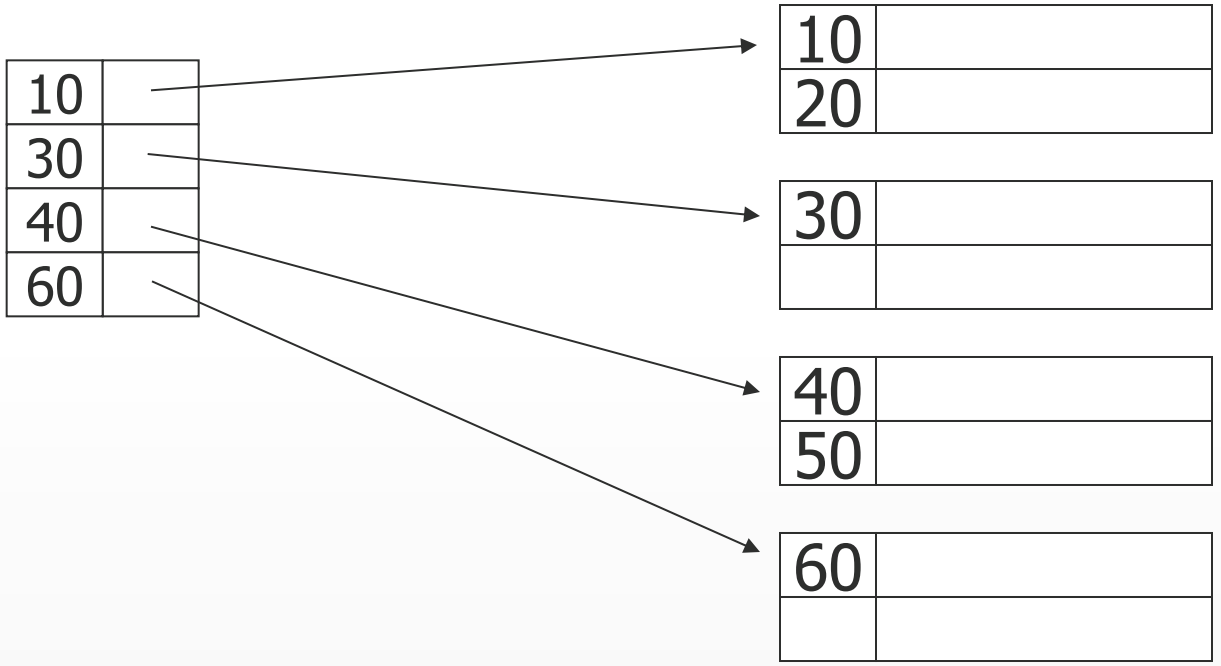
Duplikati / redak



Duplikati / redak

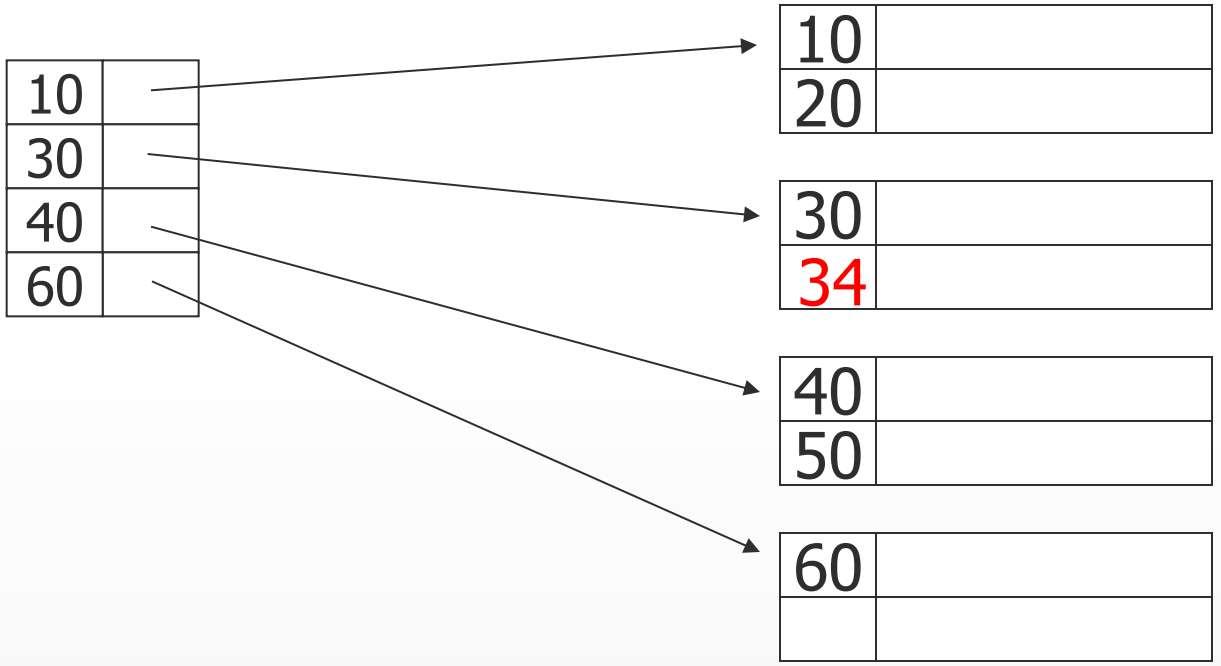


Dodavanje redak indeks



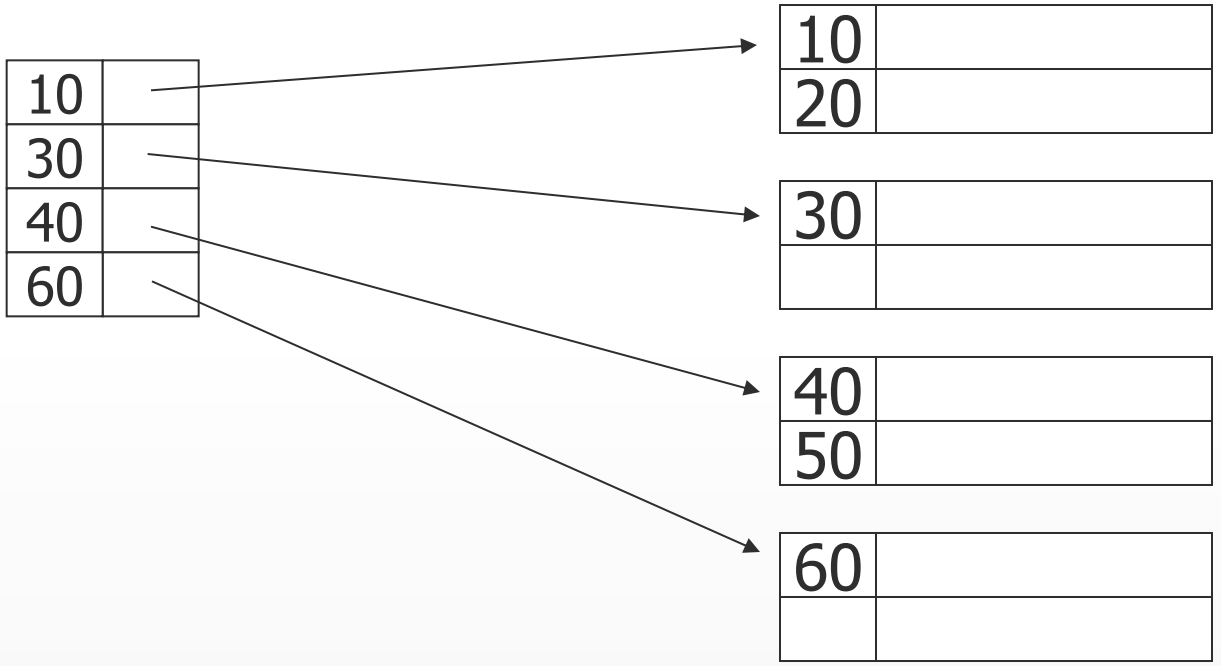
–insert record 34

Dodavanje redak indeks



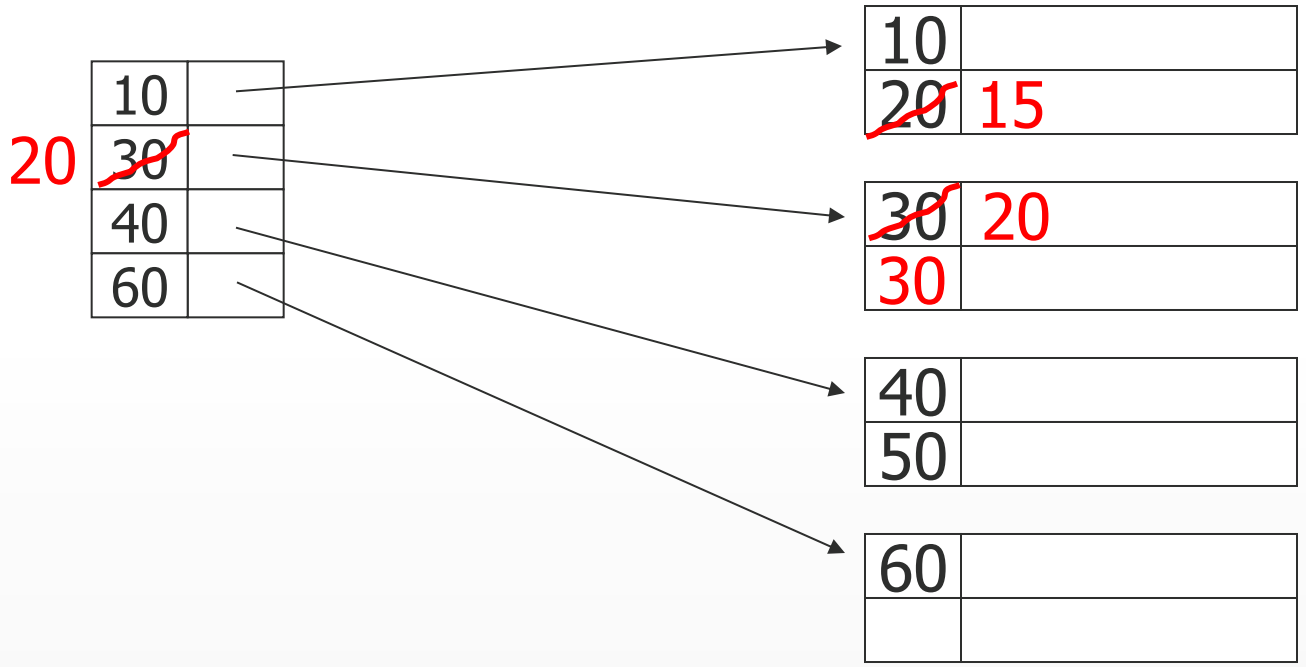
–insert record 34

Dodavanje redak indeks



–insert record 15

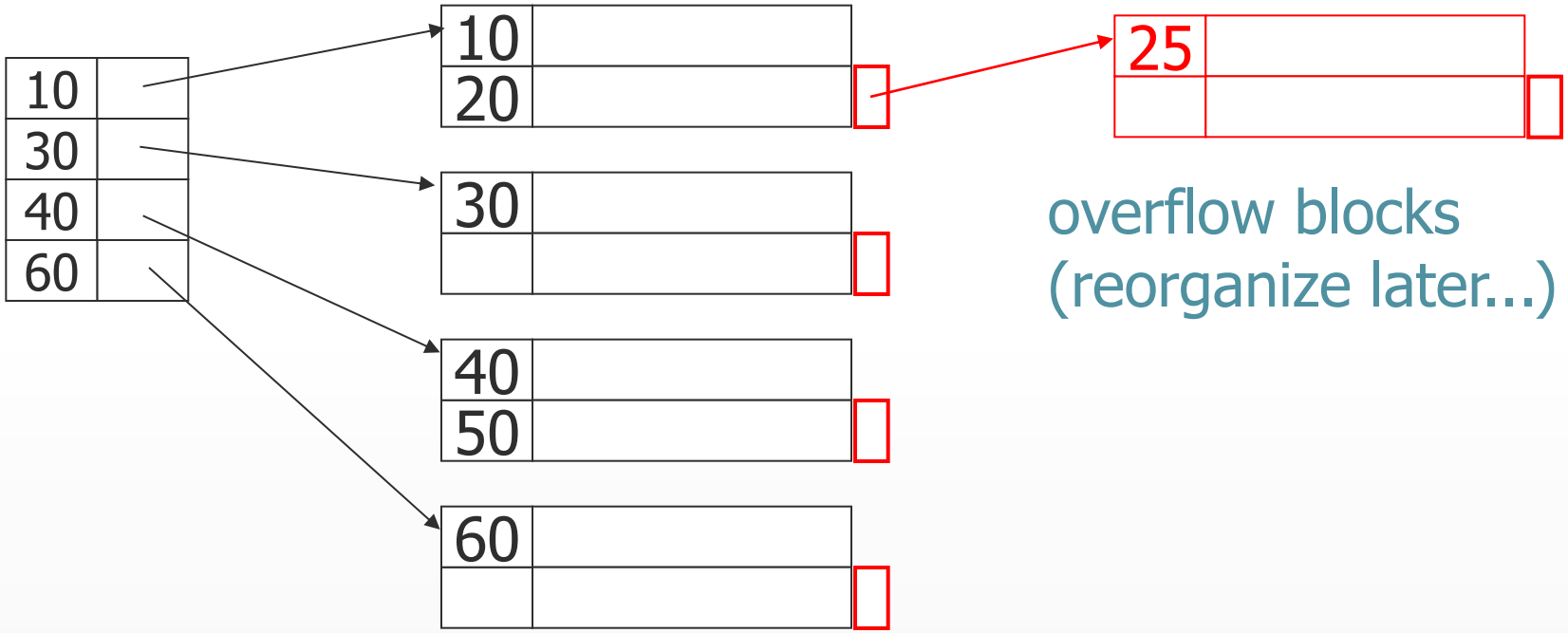
Dodavanje redak indeks



- Illustrated: Immediate reorganization
- Variation:
  - insert new block (chained file)
  - update index

–insert record 15

Dodavanje redak indeks



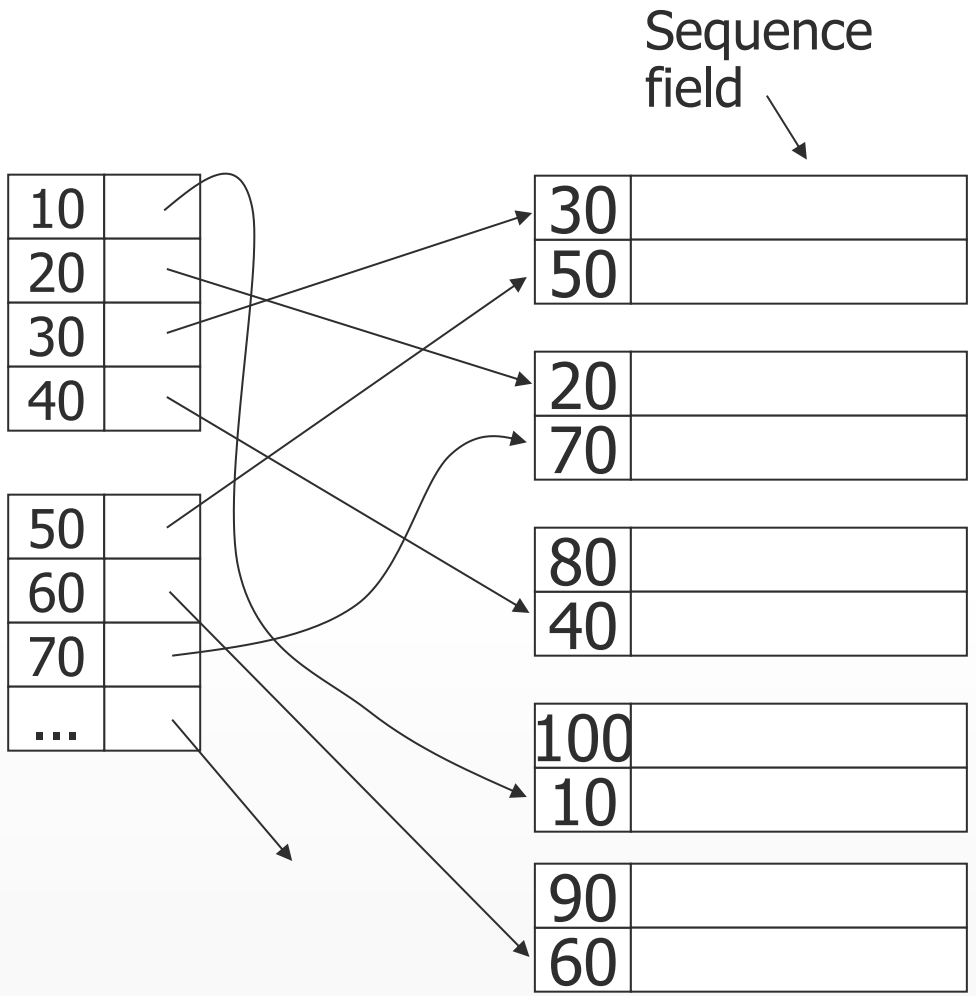
–insert record 25

Dodavanje gust indeks

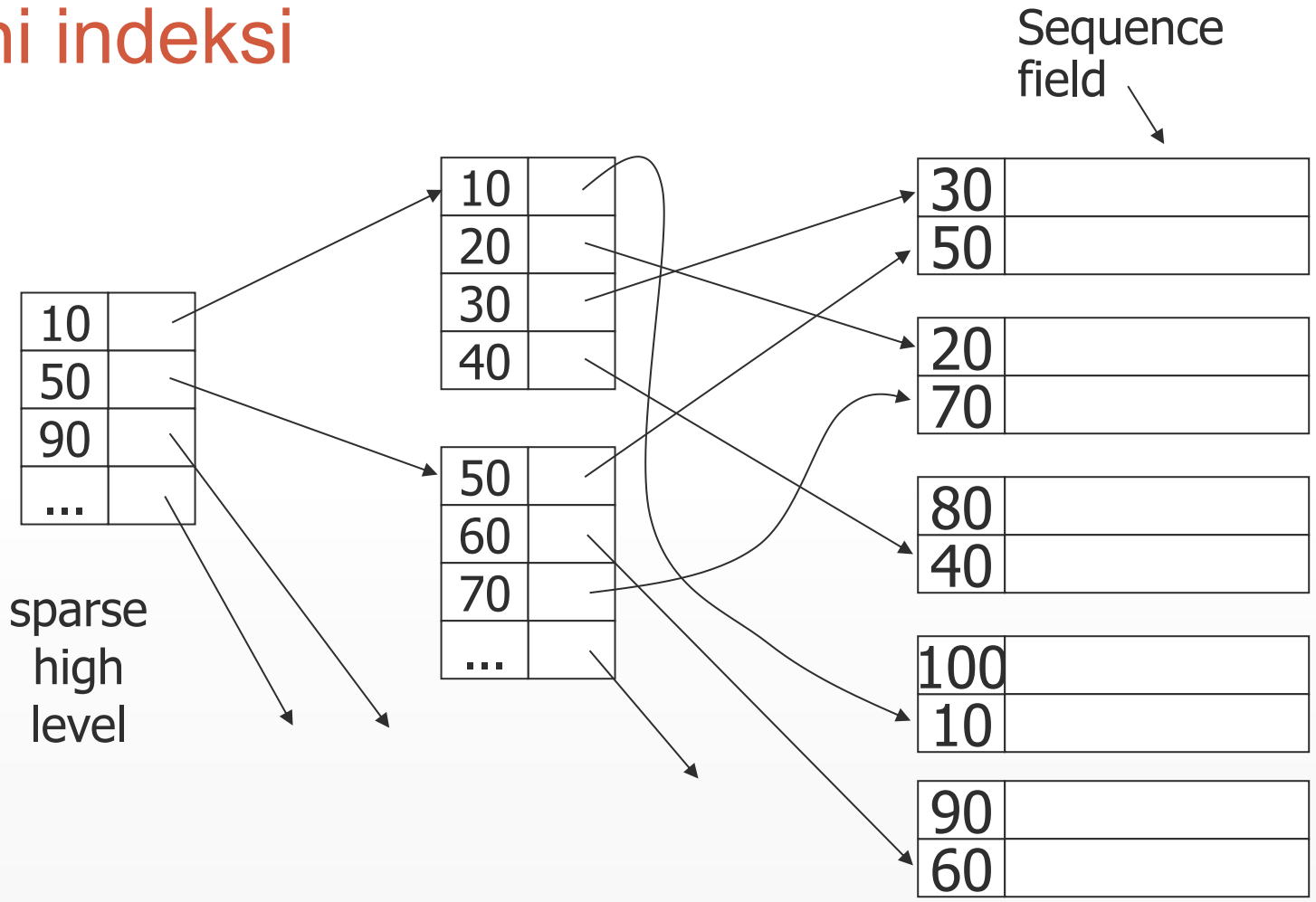
Analogno retkom

Sekundarni indeksi

- Indeksirano polje ne odgovara kriterijumu uređenja fajla sa podacima.
- Indeks mora biti gust.
- Pointer u slogovima indeksa mora biti pointer ka slogu podataka.



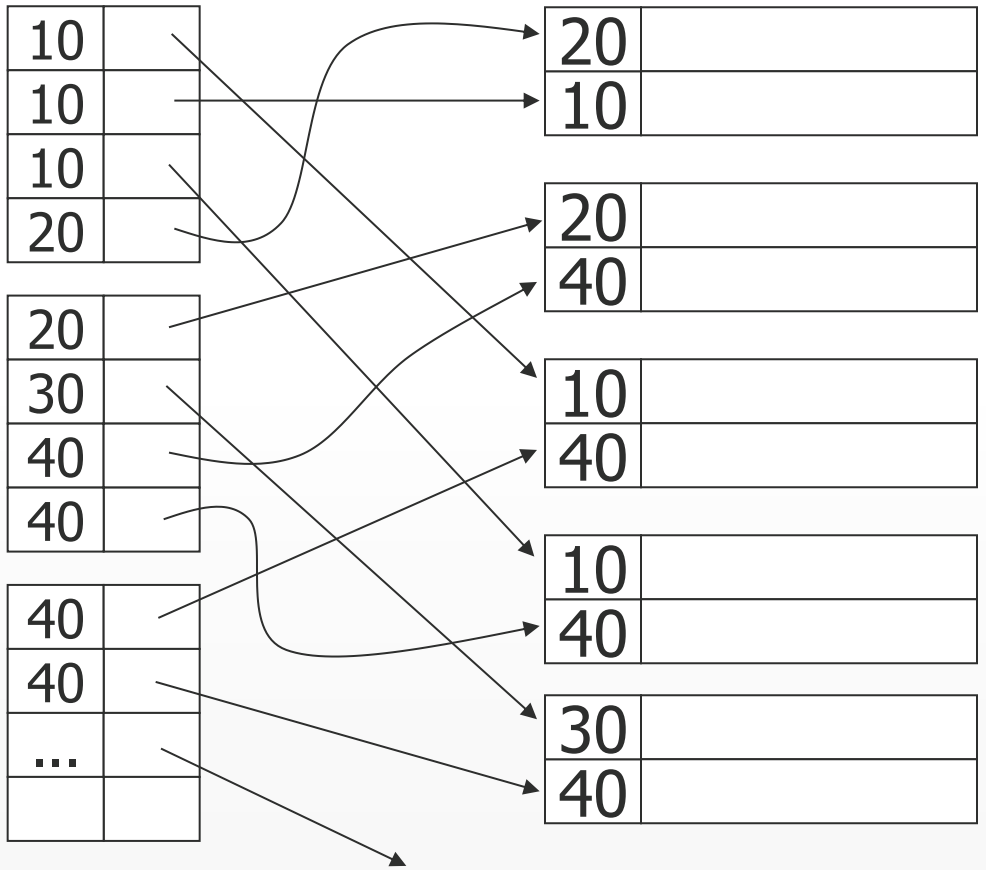
Sekundarni indeksi



Duplikati i sekundarni indeksi

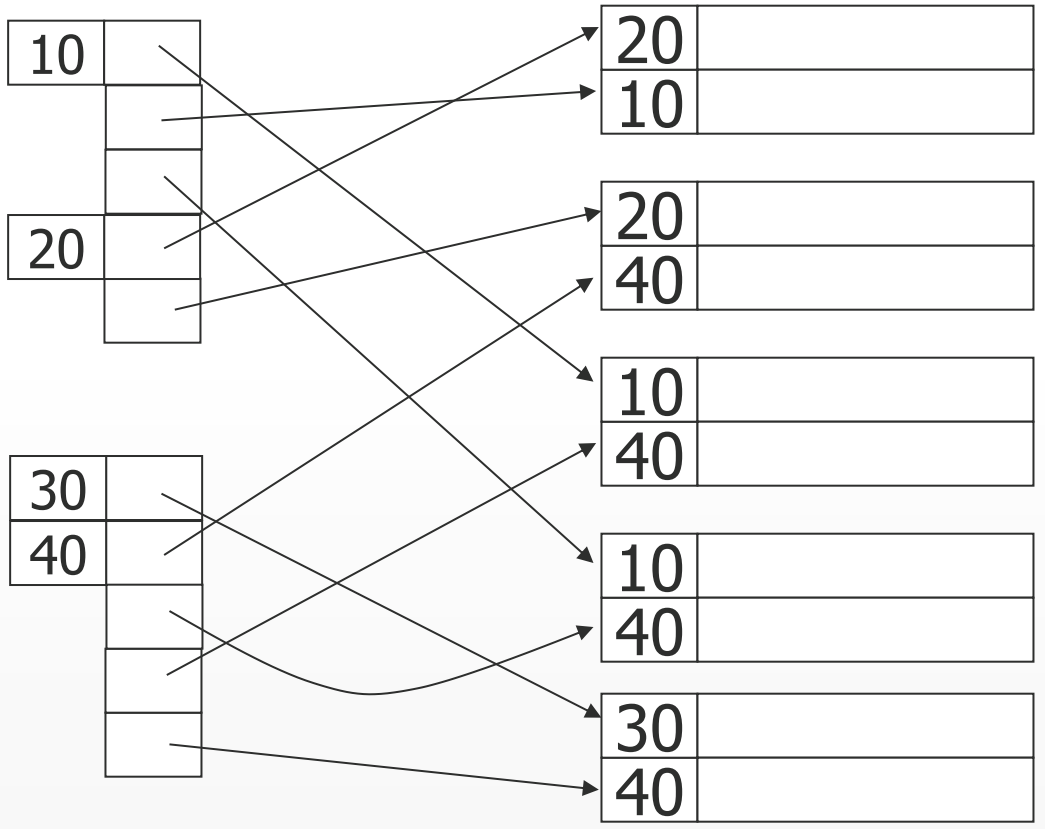
Problem:
excess overhead!

- disk space
- search time

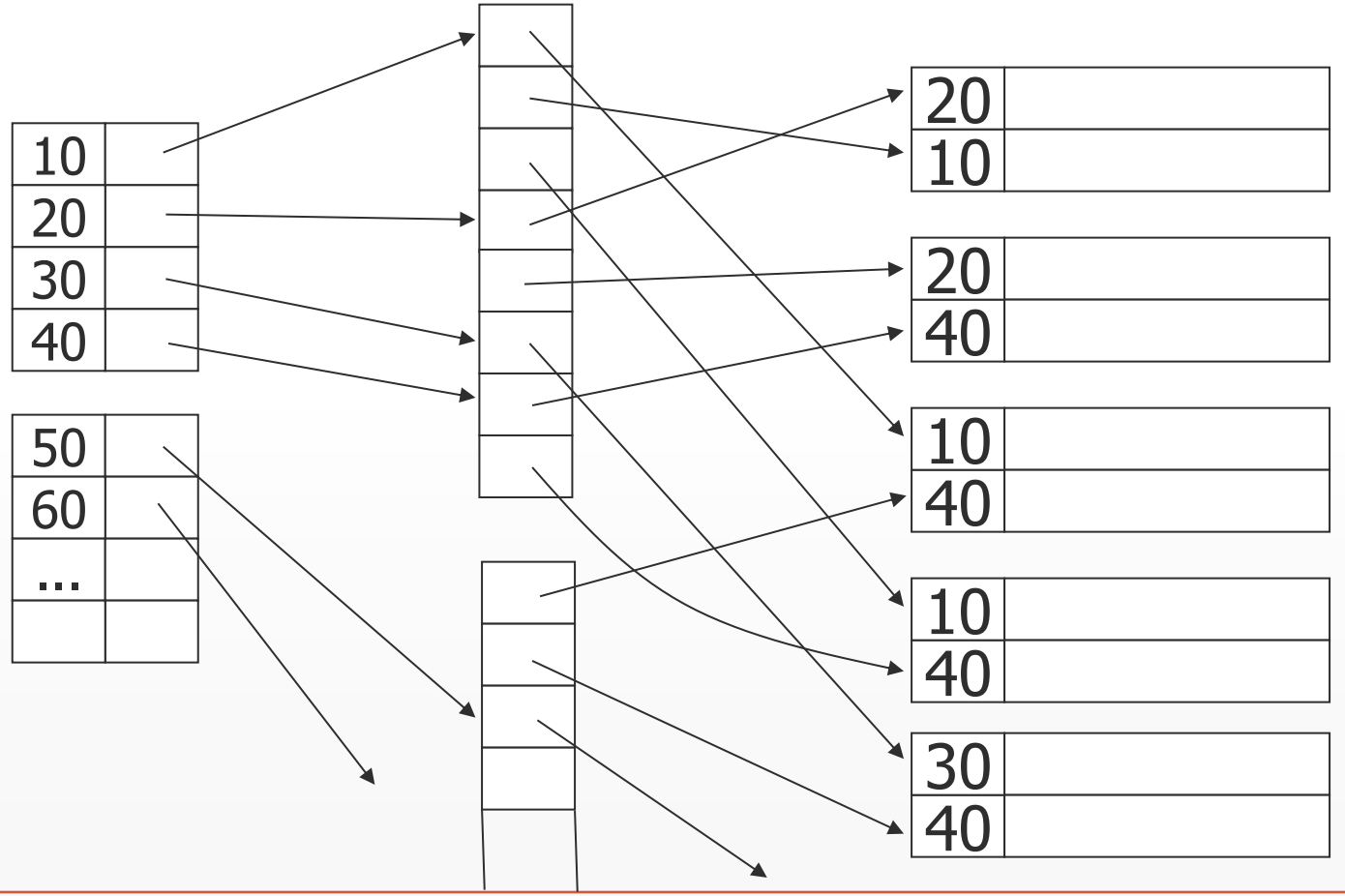


Duplikati i sekundarni indeksi

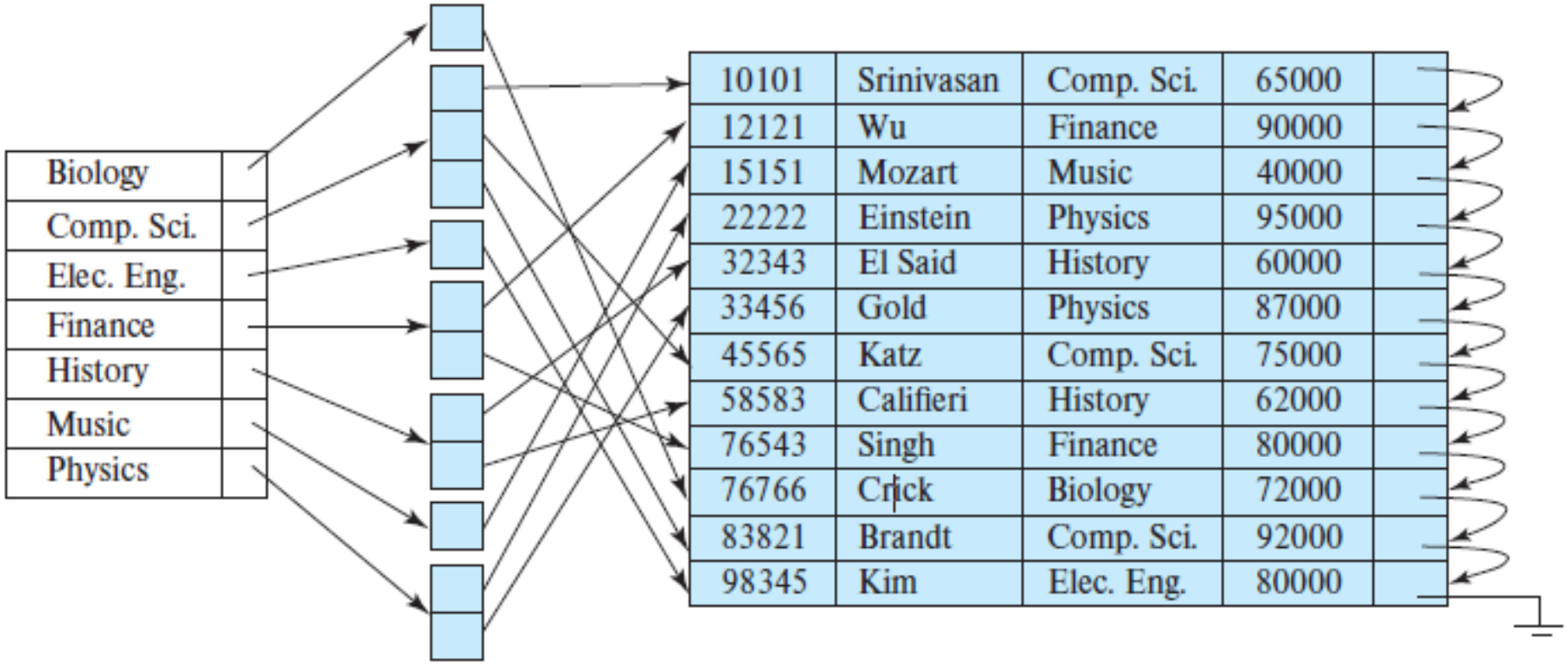
Problem:
variable size records in index!



Duplikati i sekundarni indeksi

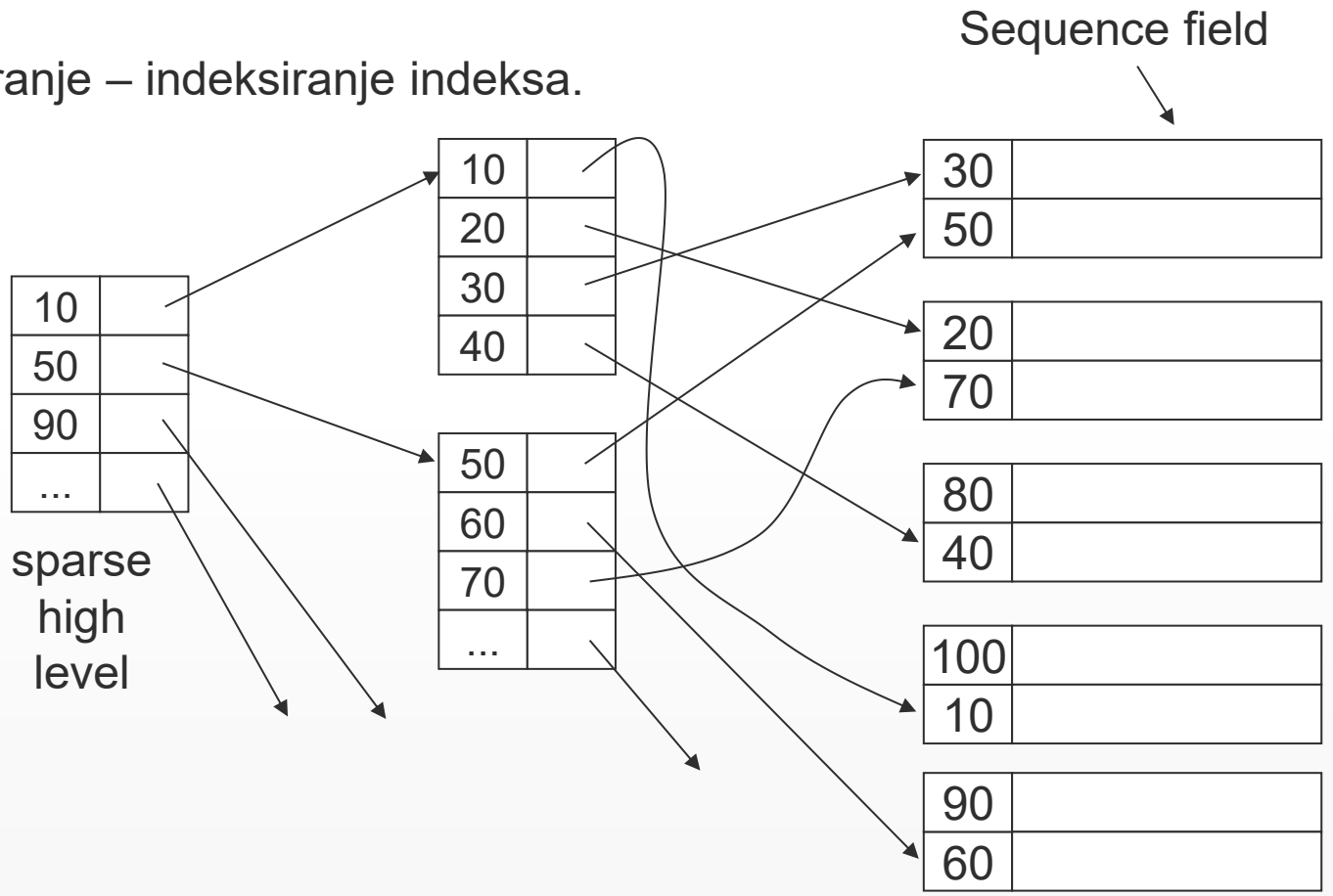


Duplikati i sekundarni indeksi



Višenivoiski indeksi

- Višenivoisko, hijerarhijsko indeksiranje – indeksiranje indeksa.



B+ stablo

Uređeni indeksi

Implementacija B+ stabla kao uređenog indeksa

Implementacija uređenih indeksa - B+ stablo

- B+ stablo pretrage je vrsta grafa koja se najčešće koristi u RDBMS-ovima kao struktura za implementaciju uređenih indeksa.
- Pripada familiji B stabala.
  - B Tree (1971) – svaki čvor ima više dece, def. minimum i maksimum broja dece, balansiran, u čvorovima podaci/celi slogovi
  - B+ Tree (1973) – B stablo u kome se pokazivači na podatke ili sami podaci nalaze samo u listovima stabla.
  - B* Tree (1977) – Svaki čvor popunjen najmanje 2/3, koren ima najmanje 2 neposredna sledbenika, deljenje čvorova samo ako ne može da se izvrši prelivanje na rođake sa istog nivoa
  - B^{link}Tree (1981) – listovi međusobno spregnuti u dvostruko povezanu listu.
- U nastavku će biti predstavljena DBMS implementacija B+ stabla, koja predstavlja mešavinu raznih verzija B-stabla, ali se u literaturi najčešće koristi termin B+ stablo.

B+ stablo pretrage

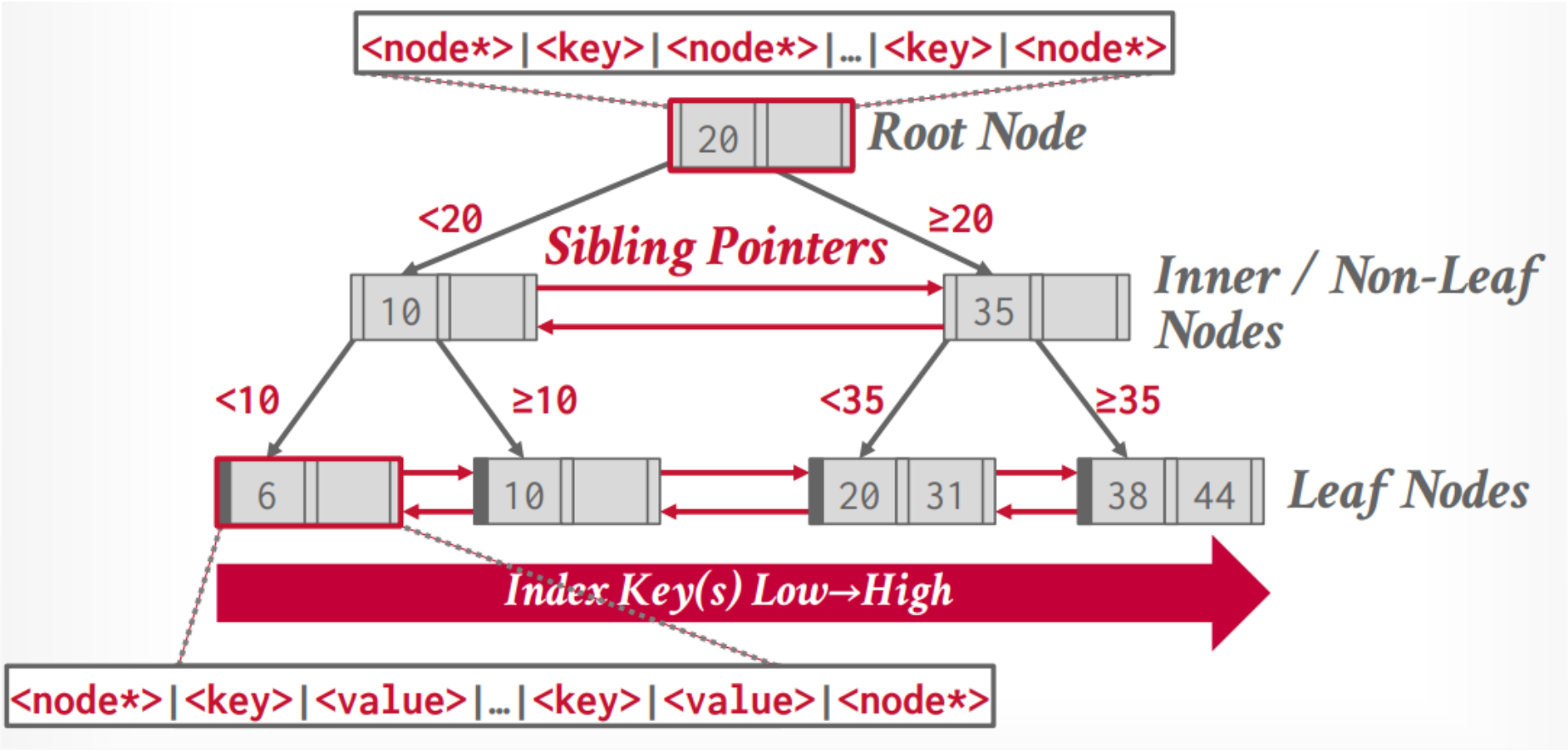
- **B+ stablo**
  - Dinamička struktura podataka
  - Balansirano stablo pretrage – svi listovi su na istom nivou, a samo stablo uređeno (vrednosti svih elemenata u levom podstablu su manje od vrednosti čvora sa kojom su “vezani”)
  - Stepena m, reda d

    Stepen* stabla m - maksimalan broj potomaka svakog čvora (matematički - maksimalan izlazni stepen čvora je m), maksimalan broj ključeva je m-1

    Red* d - minimalan broj ključeva u svakom čvoru, sa izuzetkom korena (matematički – minimalni izlazni stepen čvorova je d+1). Čvorovi moraju biti bar do pola puni.
  - Listovi su međusobno su spregnuti pokazivačima.
  - Pristup, dodavanje i brisanje imaju logaritamsku kompleksnost
  - Optimizovan za čitanje/pisanje velikih blokova podataka

* Za potrebe ovih predavanja uvodimo ovakva značenja. Red i stepen se razlikuju od uobičajenih u matematičkoj teoriji. Dodatno, terminologija varira i u računarskoj literaturi.
* Broj potomaka čvora – *fan-off*

B+ indeksno stablo



Koren i unutrašnji čvorovi

- Čvorovi su DB strane (ukoliko je u pitanju indeksni fajl, a ne indeksna struktura privremeno kreirana u radnoj memoriji)
- Sve **vrednosti ključeva** u svakom čvoru su **uređene**.
- Koren i unutrašnji čvorovi sadrže do m-1 ključeva i m pokazivača

$P_1 \mid K_1 \mid \ldots \mid P_{n-1} \mid K_{m-1} \mid P_m$

Ki - vrednosti indeksnog polja.

Pi - pokazivači na potomke

- Čvor sa k ključeva ima k+1 ne-null potomka

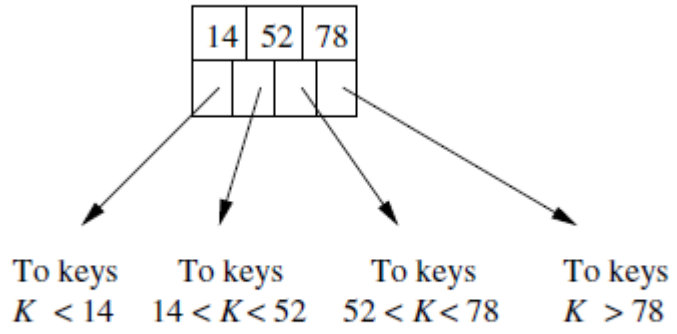


Figure 12: A typical interior node of a B-tree

Listovi

Teorijski, imaju m-1 par <Kljuc, Vrednost> i jedan pokazivač na susedni list.

Vrednost u paru < Kljuc, Vrednost > mogu biti:

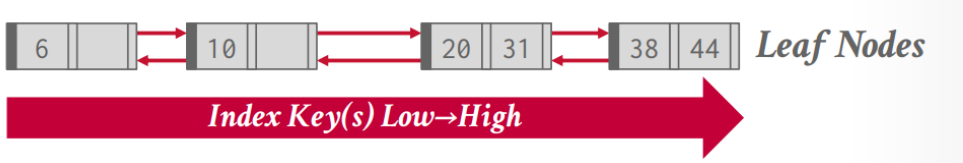
- Pokazivači na lokacije torki ili
- Kompletne torke (ređi slučaj)

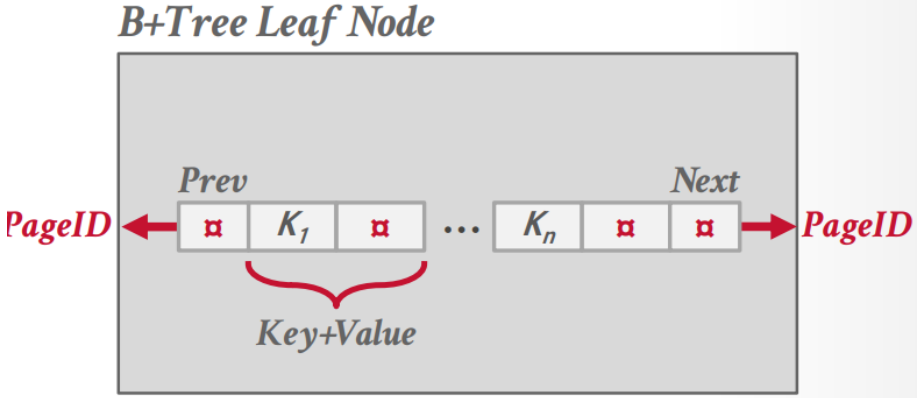
U praksi

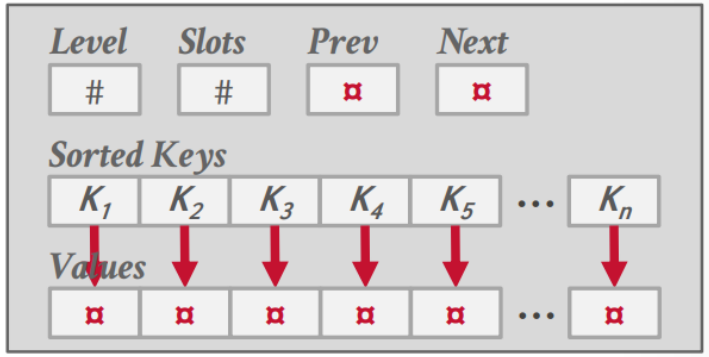
- postoji pokazivač i na prethodnika u povezanoj listi listova

<node*>|<key>|<value>|…|<key>|<value>|<node*>

- maksimalan broj parova <Kljuc, Vrednost> ne mora biti m-1, kao u unutrašnjim čvorovima. Na to utiče priroda vrednosti (da li ceo slog, da li ključevi imaju duplikate, …)







Popunjenost čvorova

- **Koren** ima najmanje 1 ključ i 2 potomka
  (osim u trivijalnim slučajevima)
- **Listovi** – najmanje polovina parova ključ/vrednost je ‘upotrebljena’
  Minimalan broj ključeva je

  $\left\lceil \frac{m-1}{2} \right\rceil$, m - stepen stabla

  Za m = 5, to je 2. Za m = 4, takođe 2.
- **Unutrašnji čvorovi –** pokazivači najmanje do pola ‘puni’
  Najmanje polovina pointera imaju nenul vrednost i ukazuju na potomke, tj. blokove na sledećem nivou u stablu.
  $\left\lceil \frac{m}{2} \right\rceil$ <= izlazni_stepen_čvora <= m
- d <= #broj kljuceva <= 2d (odnosno, m-1)

Dodavanje

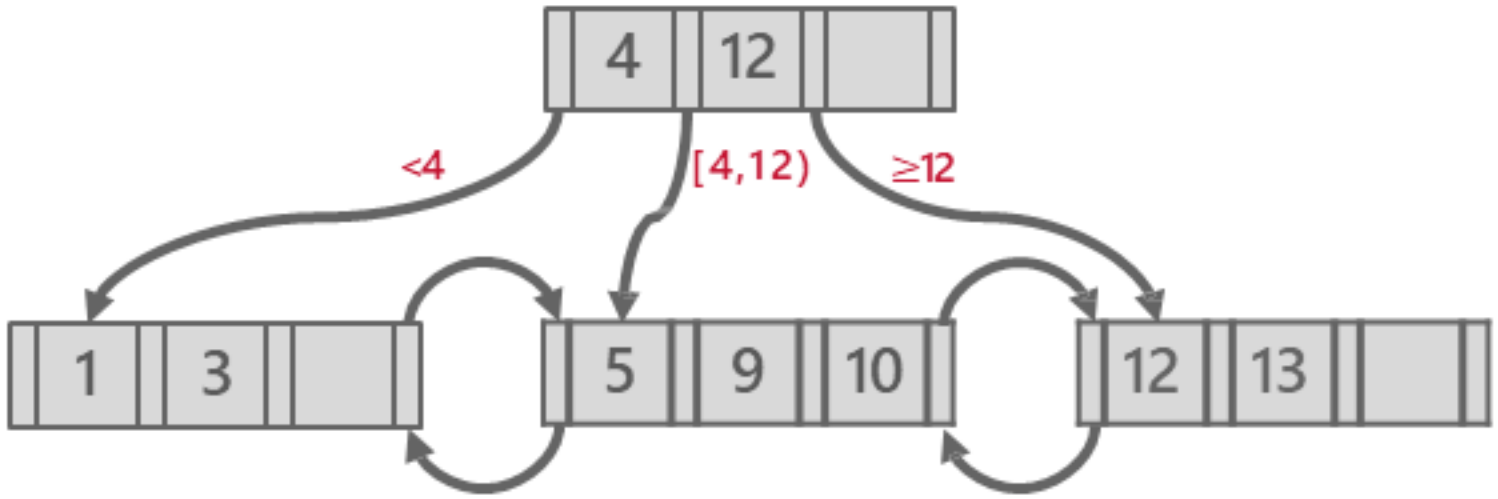
- Da bi se novi unos umetnuo u B+ stablo, potrebno je proći kroz stablo od vrha ka dnu i koristiti unutrašnje čvorove da bi se odredilo u koji list treba umetnuti ključ.

Algoritam

- Pronalazi se odgovarajući list L.
- Novi unos se dodaje u L, čuvajući uređenje
  - Ako L ima dovoljno prostora, operacija je završena.
  - U suprotnom, L se deli na dva čvora, L1 i L2. Ključevi se ravnomerno raspodeljuju, a srednji ključ se kopira nagore. U roditeljski čvor L se umeće novi unos koji pokazuje na L2.
    * Unapređenje alg. – može se raditi redistribucija ključeva između L i susednog lista
- Ako je potrebno podeliti i roditeljski čvor L (jer je pun), unosi se ravnomerno raspodeljuju, ali se srednji ključ prosleđuje nagore.

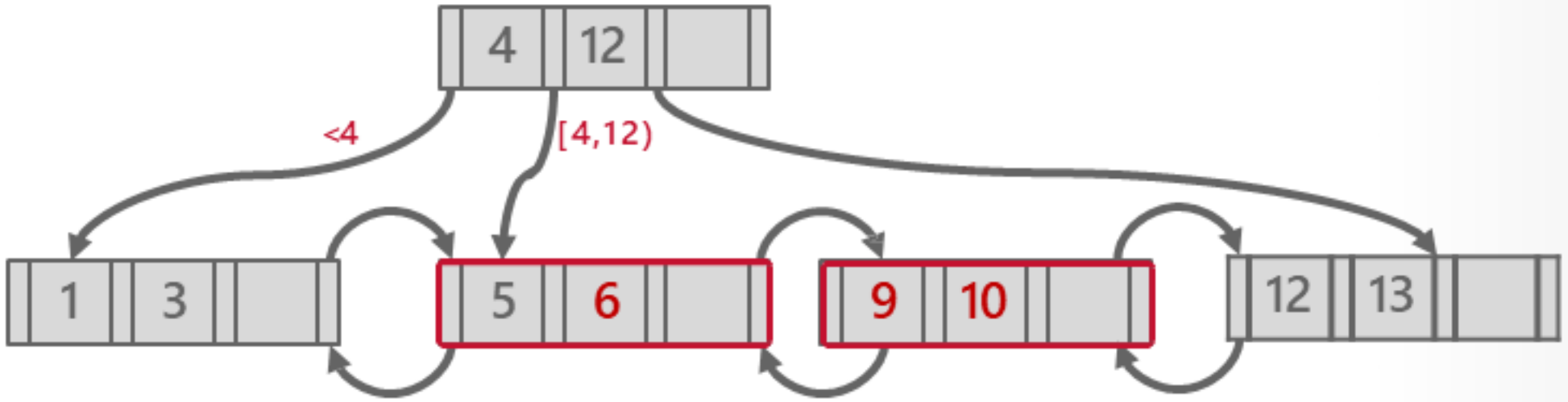
B+TREE – INSERT EXAMPLE (1)

Insert 6



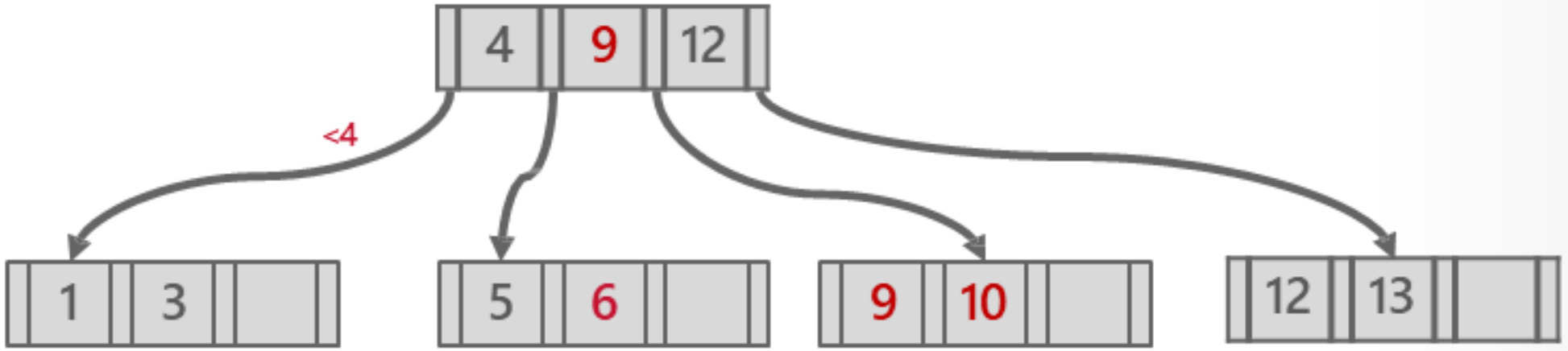
B+TREE – INSERT EXAMPLE (1)

Insert 6



B+TREE – INSERT EXAMPLE (1)

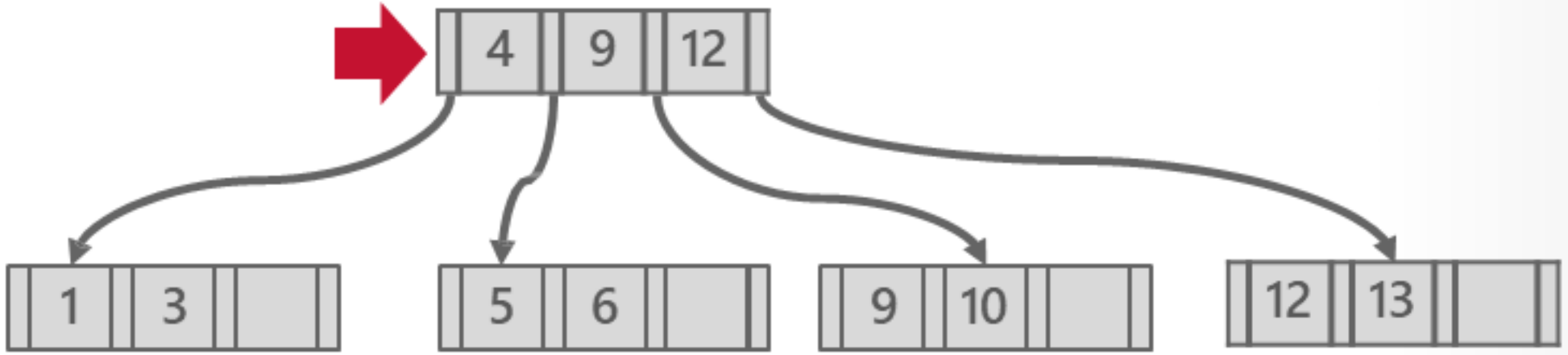
Insert 6



B+TREE – INSERT EXAMPLE (1)

Insert 6

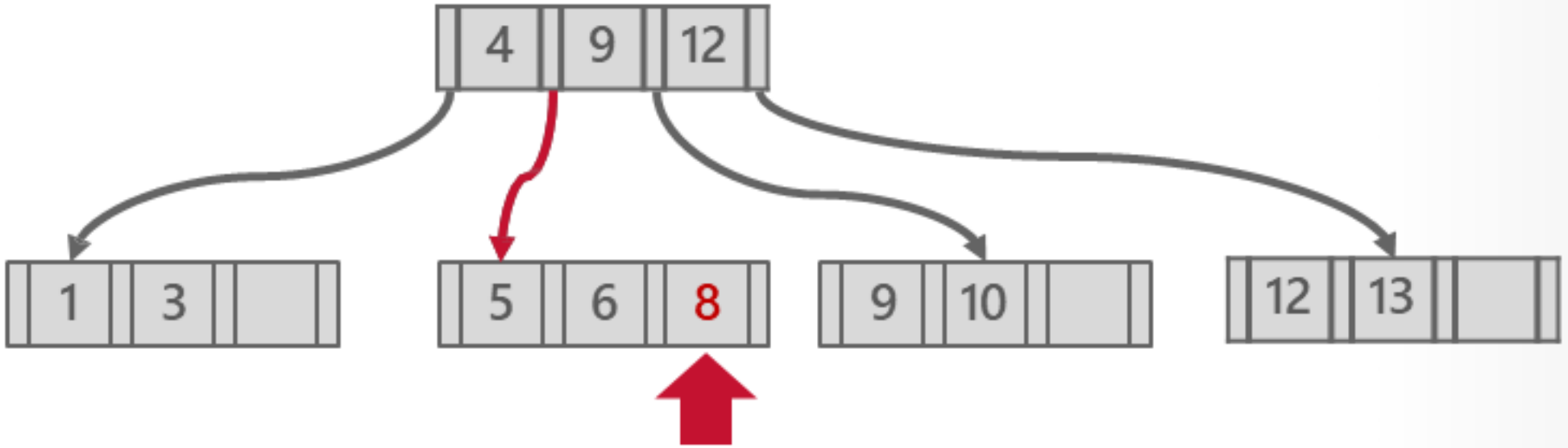
Insert 8



B+TREE – INSERT EXAMPLE (1)

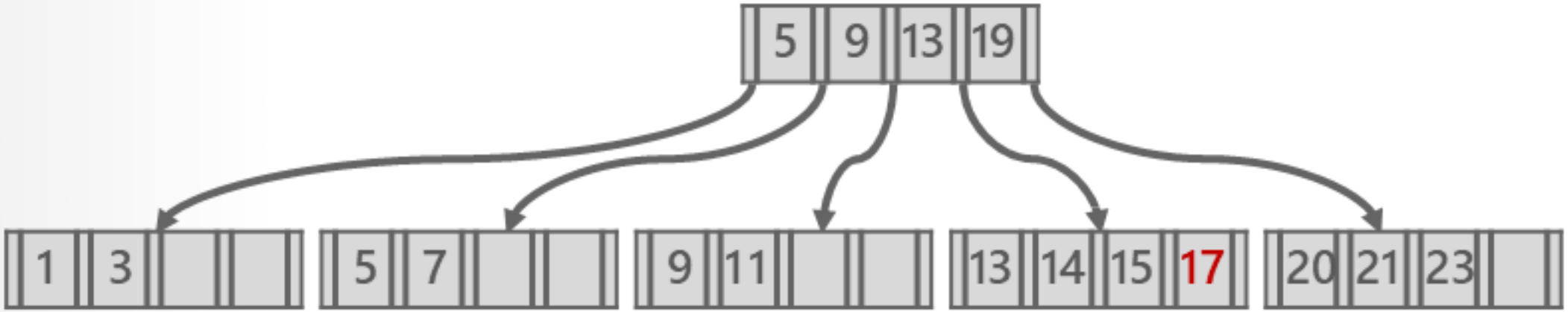
Insert 6

Insert 8



B+TREE – INSERT EXAMPLE (2)

Insert 17

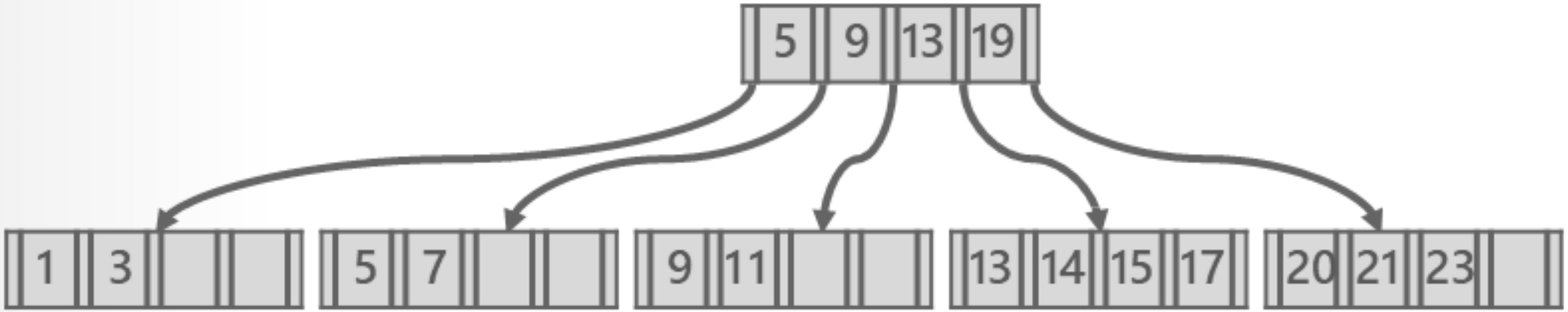


Note: New Example/Tree.

B+TREE – INSERT EXAMPLE (3)

Insert 17

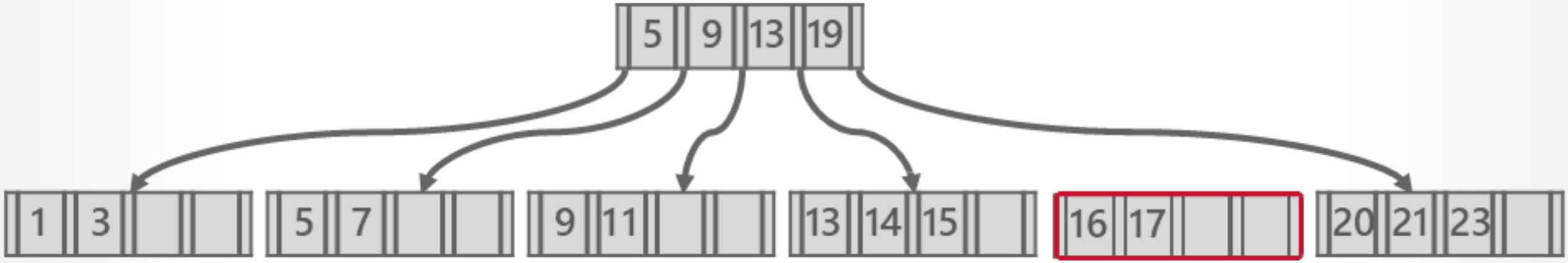
Insert 16



B+TREE – INSERT EXAMPLE (3)

Insert 17

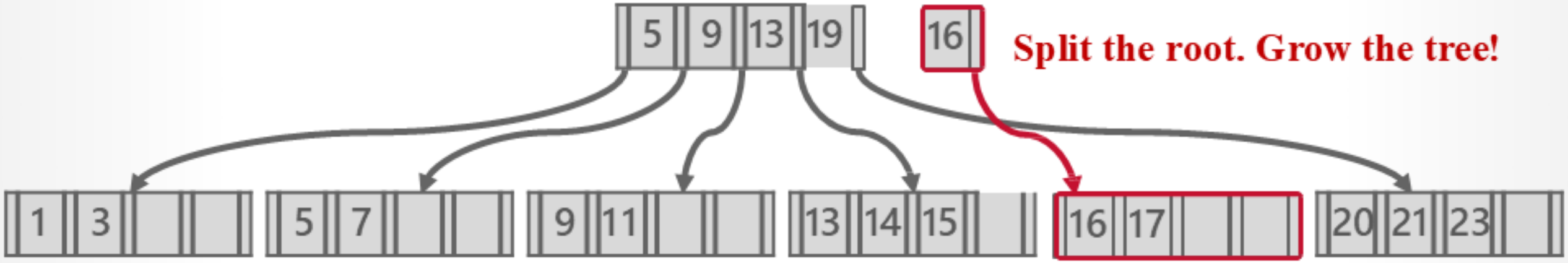
Insert 16



B+TREE – INSERT EXAMPLE (3)

Insert 17

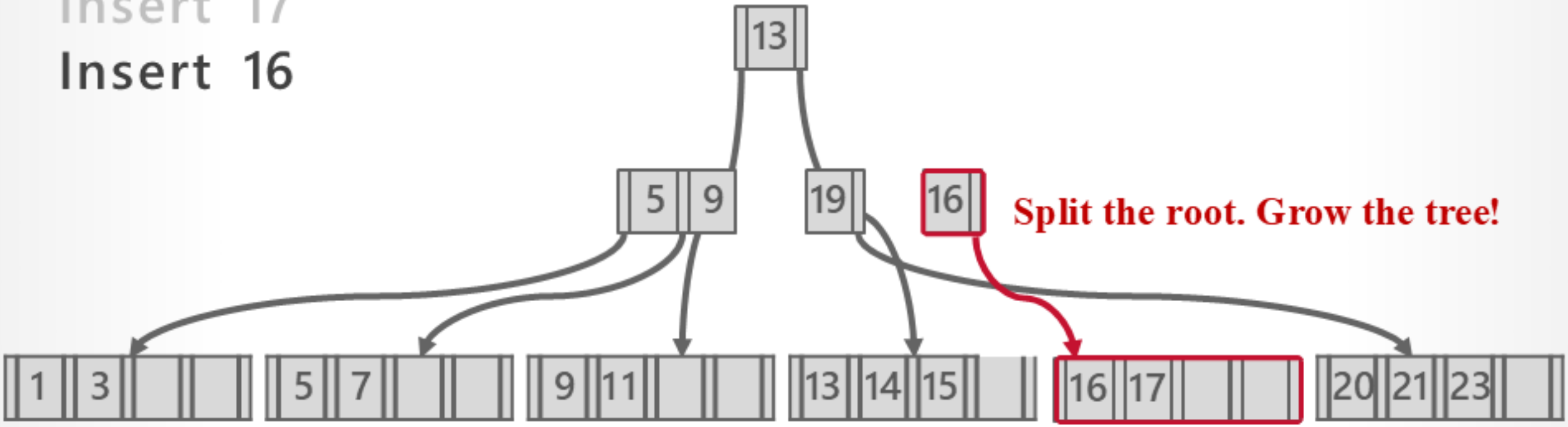
Insert 16



B+TREE – INSERT EXAMPLE (3)

Insert 17

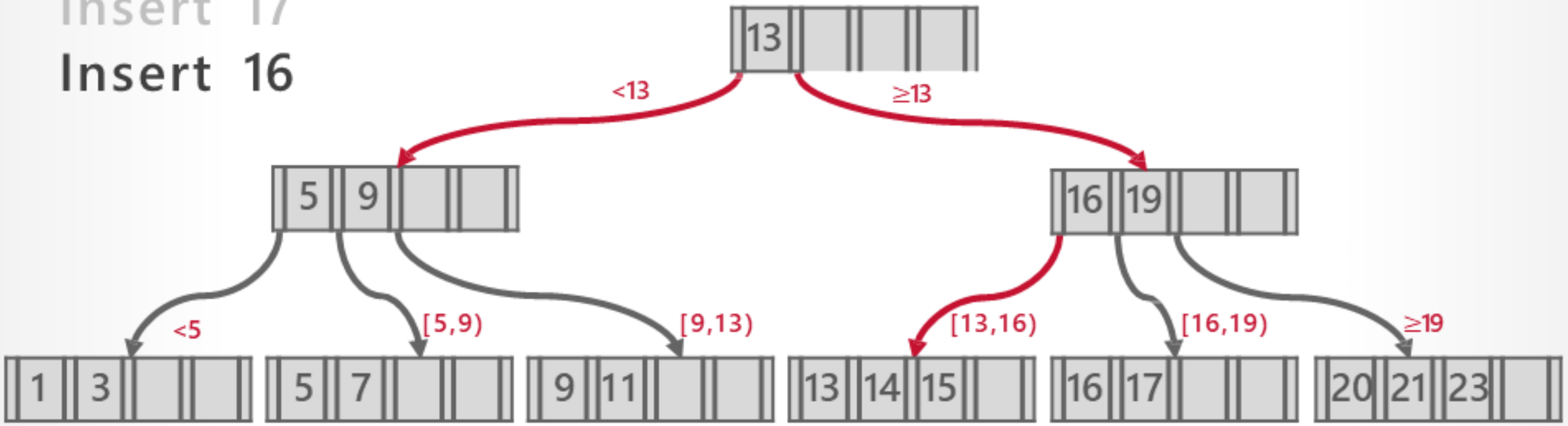
Insert 16



B+TREE – INSERT EXAMPLE (3)

Insert 17

Insert 16



Brisanje

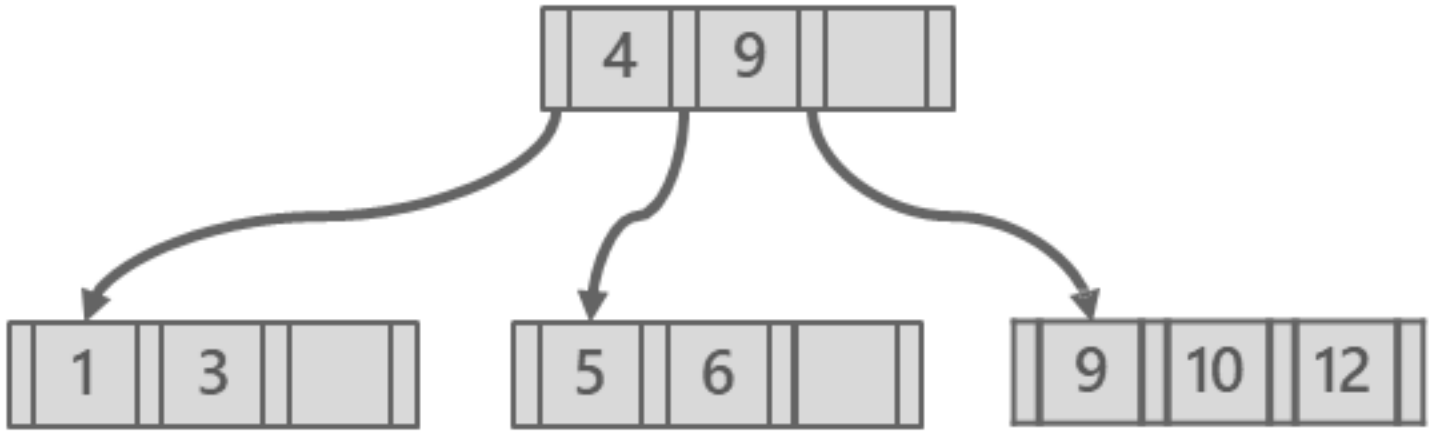
- Pri brisanju se, ukoliko čvor ostane manje od polupunog, mora izvršiti spajanje radi ponovnog balansiranja.

Algoritam

- Pronalaženje lista L u kome se nalazi vrednost koju treba izbaciti.
- Vrednost se uklanja:
  - Ako je L barem polupun, operacija je završena.
  - U suprotnom, pokušava se pozajmljivanje iz susednog čvora.
  - Ako pozajmljivanje nije moguće, vrši se spajanje L i susednog čvora.
- Ako je došlo do spajanja, par KP u roditeljskom čvoru koji pokazuje na L mora biti uklonjen.

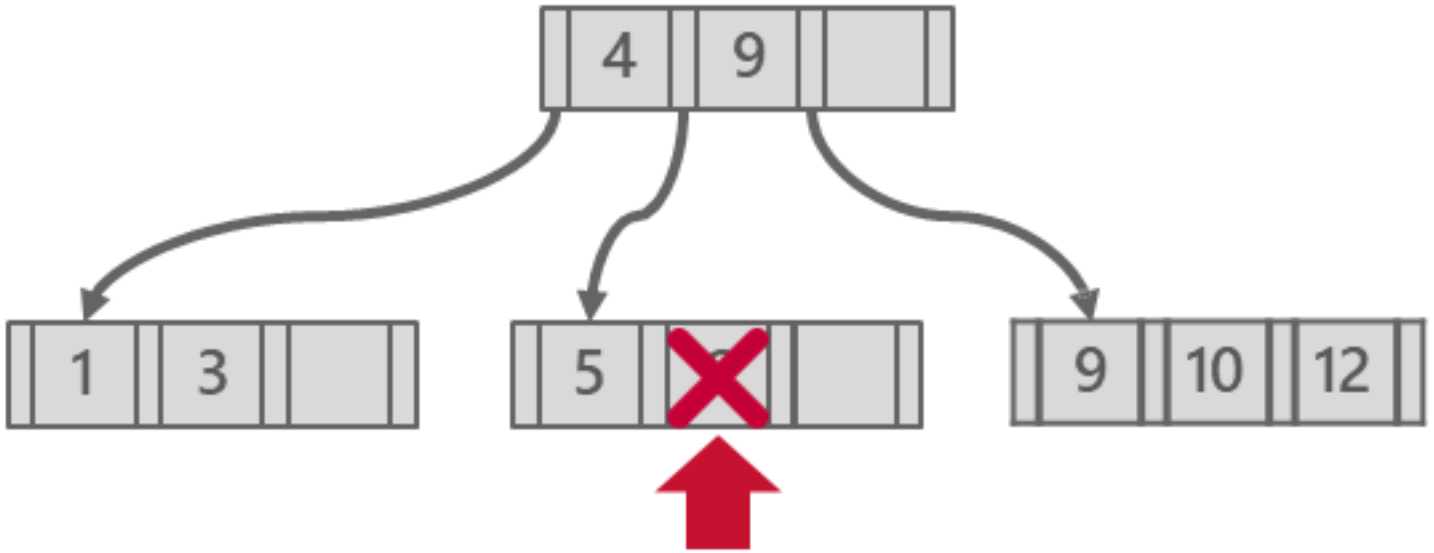
B+TREE – DELETE EXAMPLE (1)

Delete 6



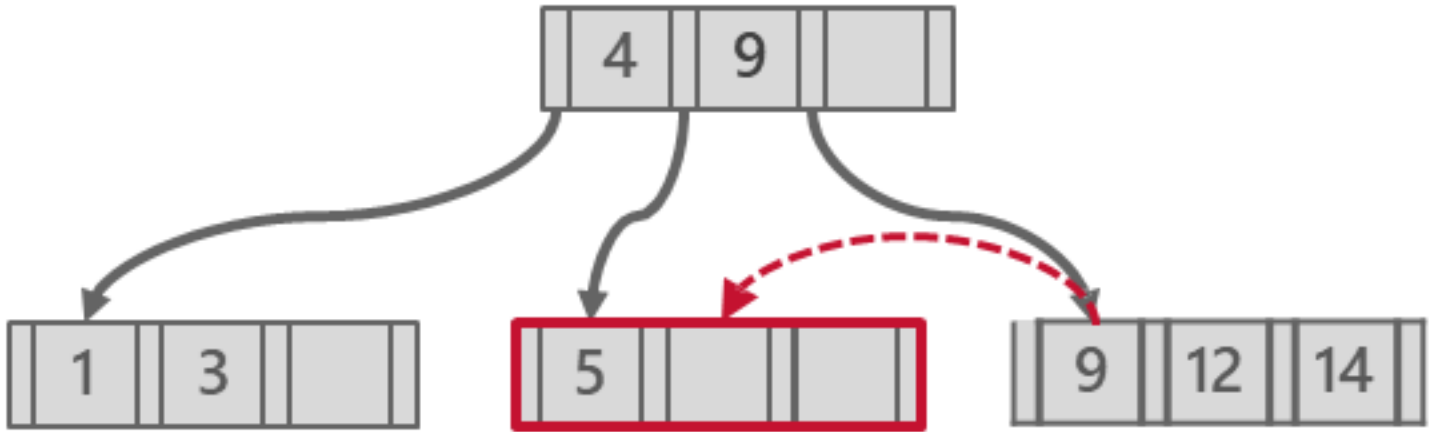
B+TREE – DELETE EXAMPLE (1)

Delete 6



B+TREE – DELETE EXAMPLE (1)

Delete 6

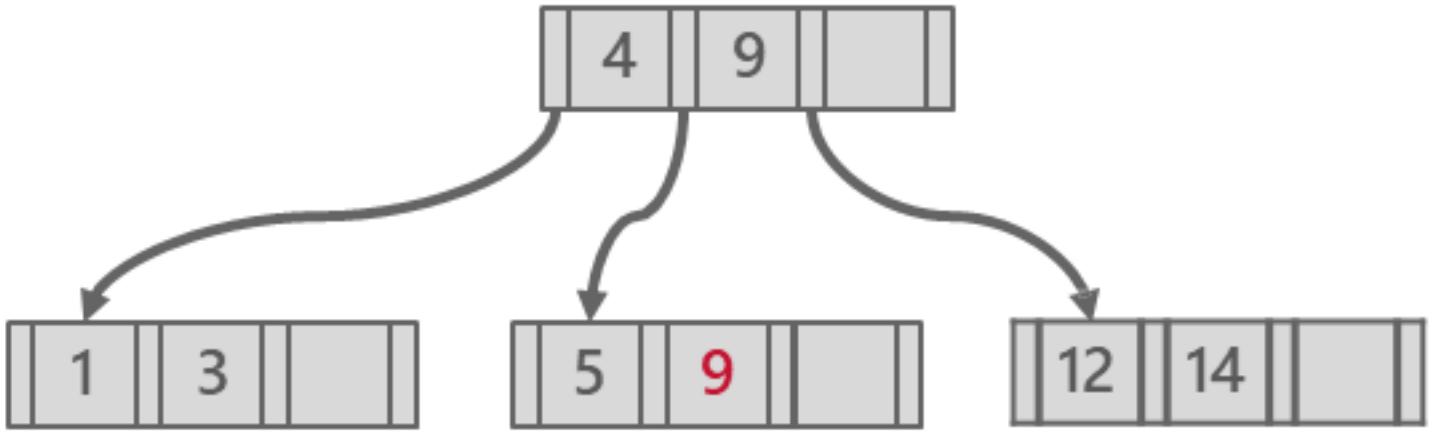


Borrow from a "rich" sibling node.

Could borrow from either sibling.

B+TREE – DELETE EXAMPLE (1)

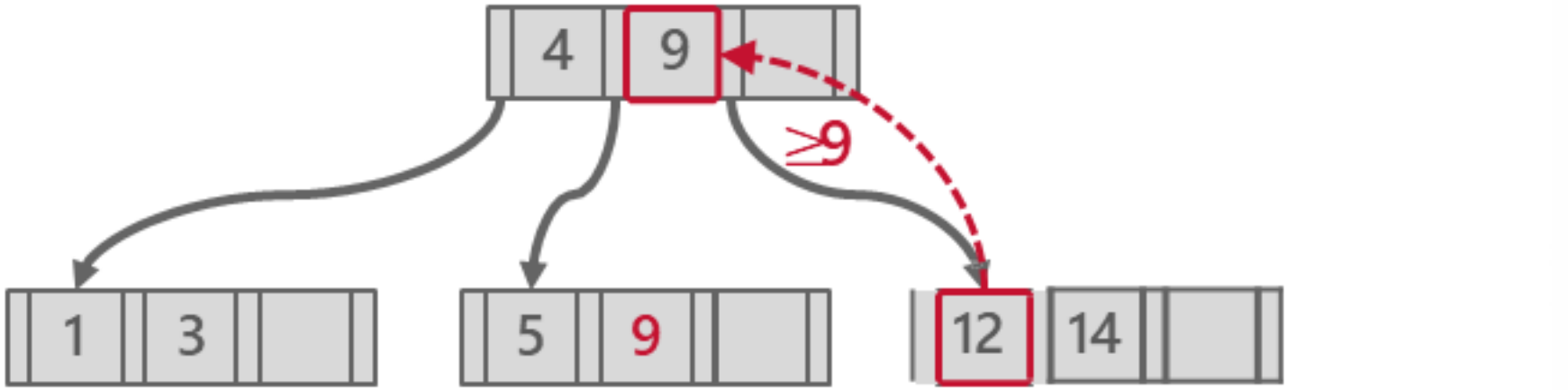
Delete 6



Need to update parent node!

B+TREE – DELETE EXAMPLE (1)

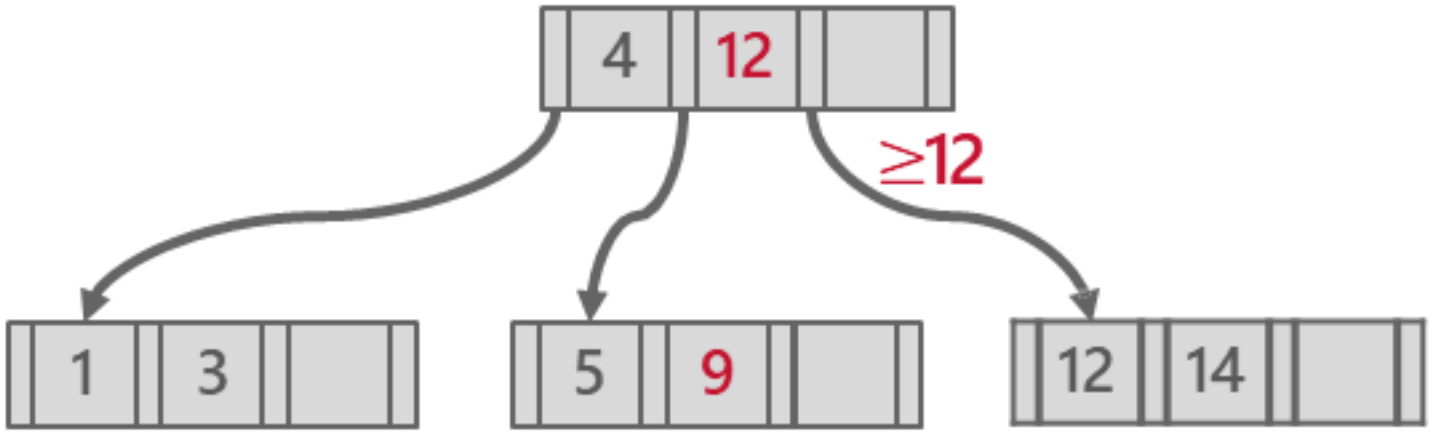
Delete 6



Need to update parent node!

B+TREE – DELETE EXAMPLE (1)

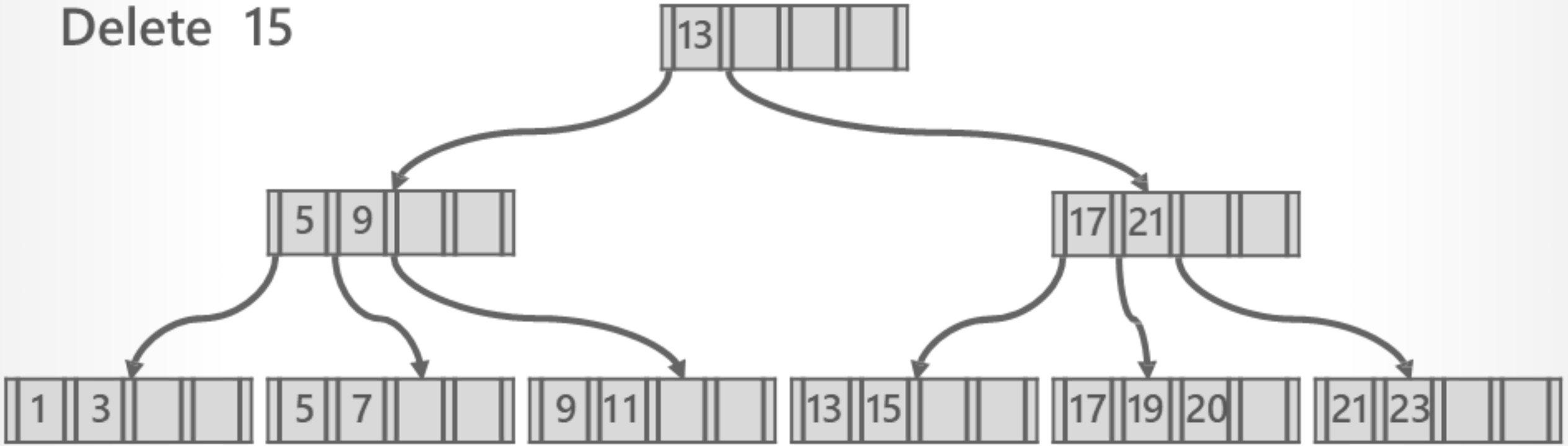
Delete 6



Need to update parent node!

B+TREE – DELETE EXAMPLE (2)

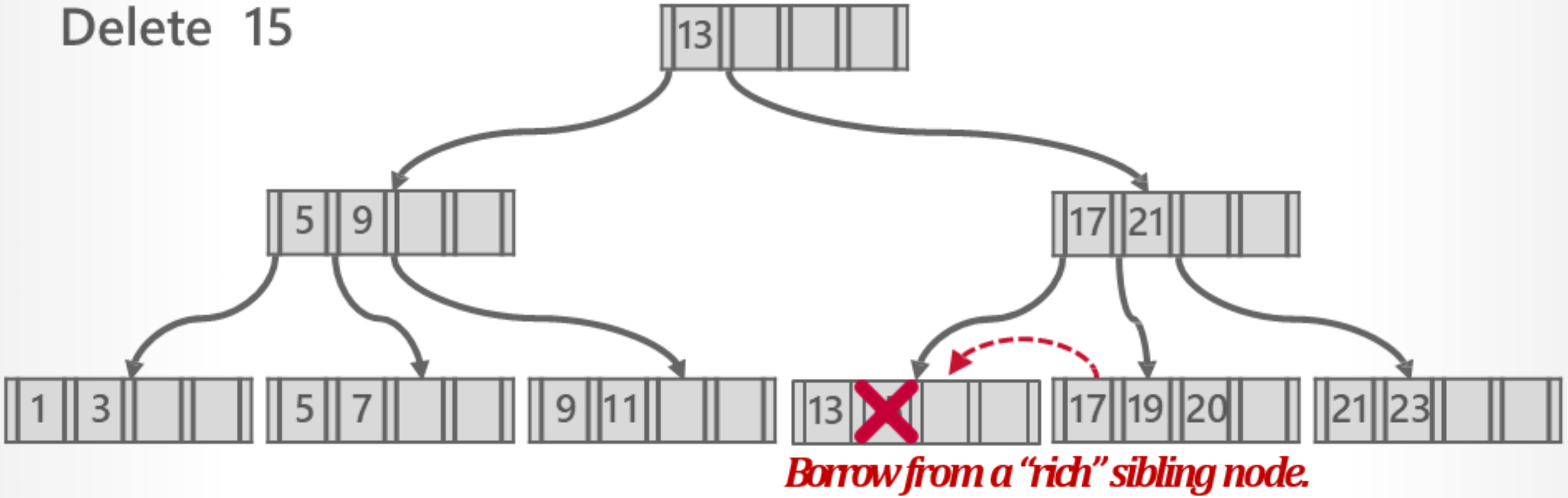
Delete 15



Note: New Example/Tree.

B+TREE – DELETE EXAMPLE (2)

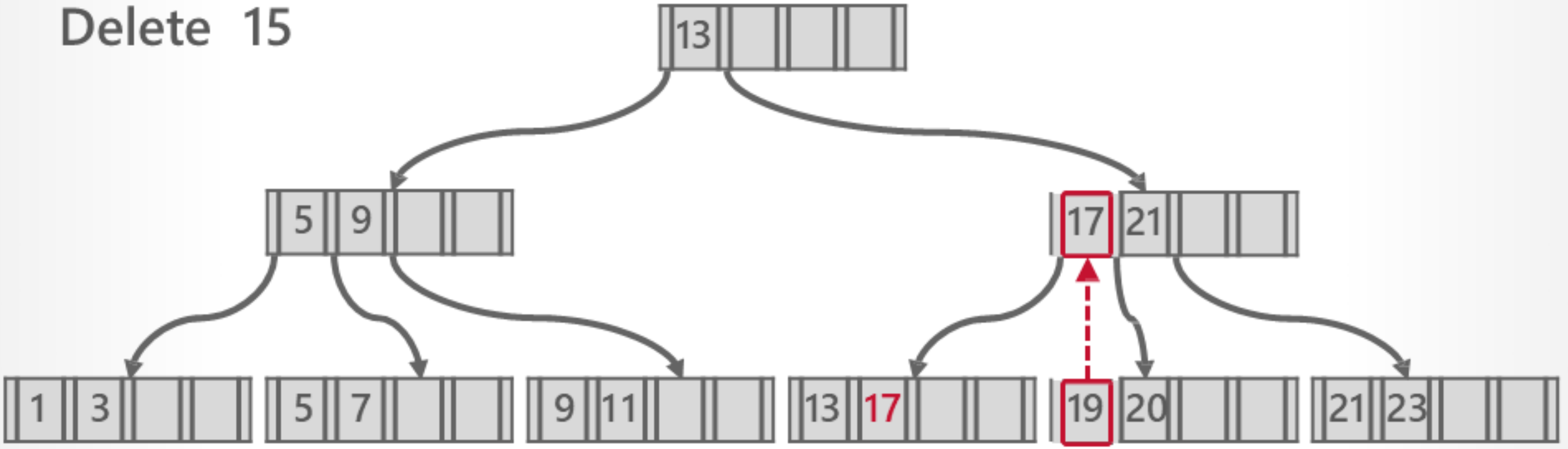
Delete 15



Note: New Example/Tree.

B+TREE – DELETE EXAMPLE (2)

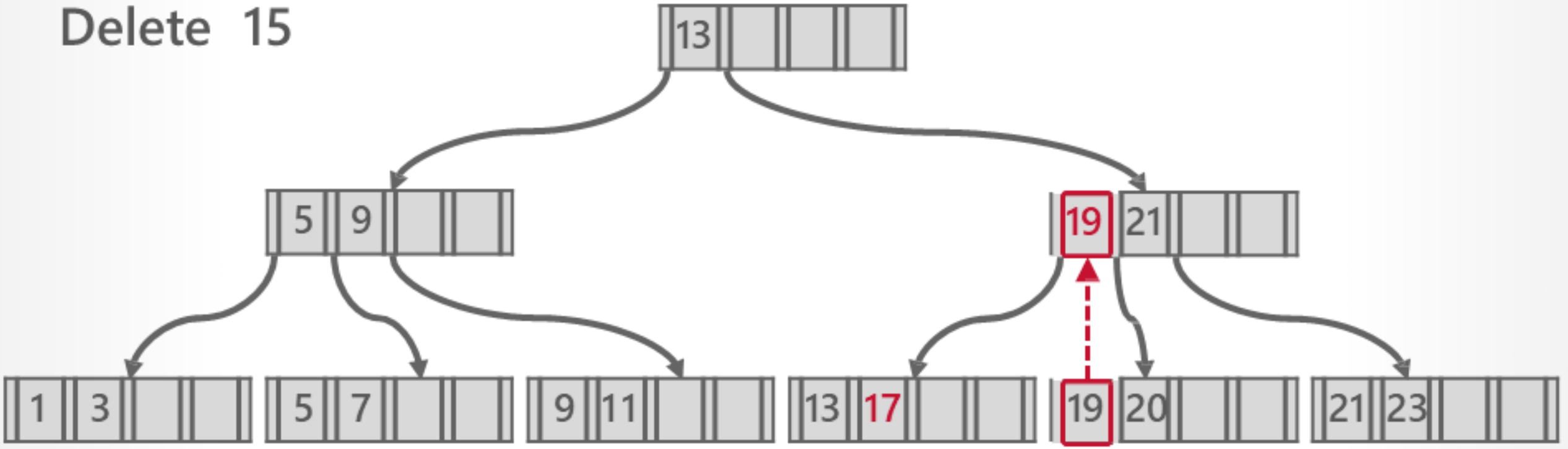
Delete 15



Need to update parent node!

B+TREE – DELETE EXAMPLE (2)

Delete 15

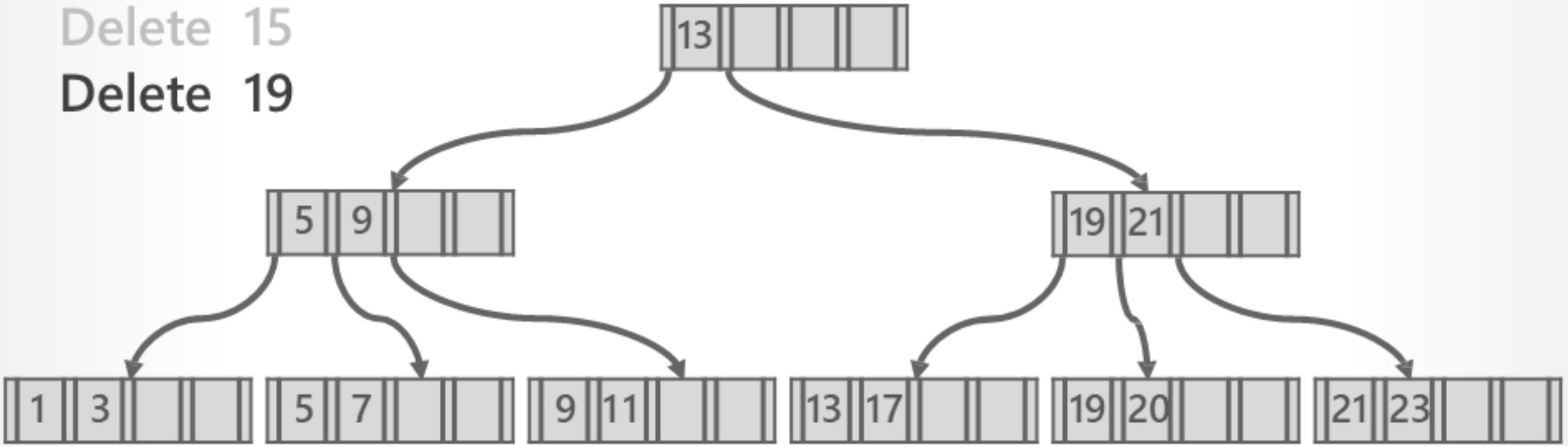


Need to update parent node!

B+TREE – DELETE EXAMPLE (3)

Delete 15

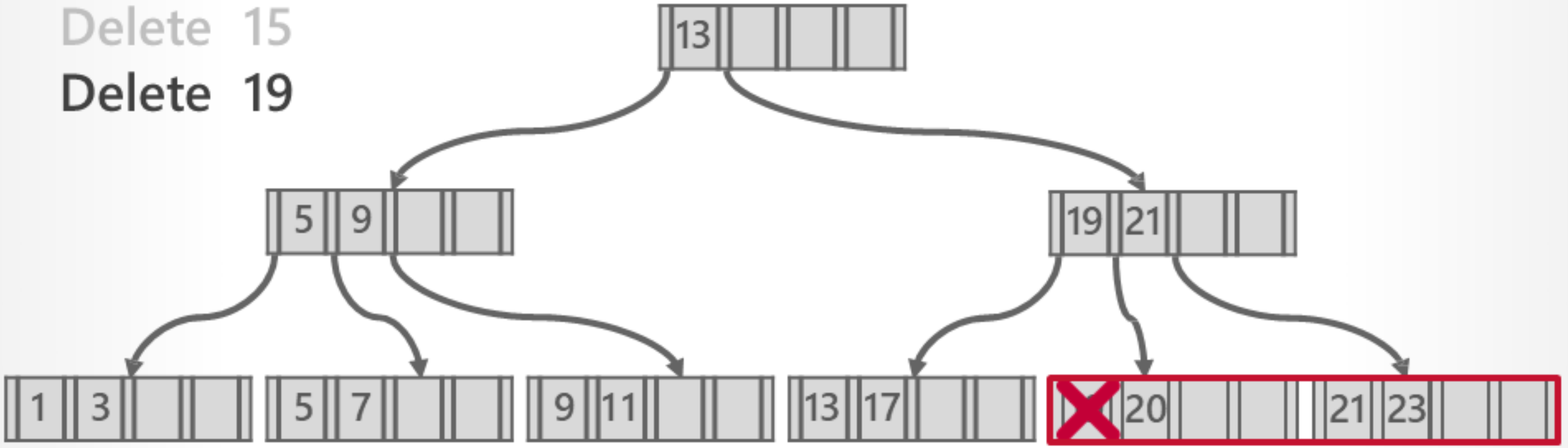
Delete 19



B+TREE – DELETE EXAMPLE (3)

Delete 15

Delete 19



Under-filled!

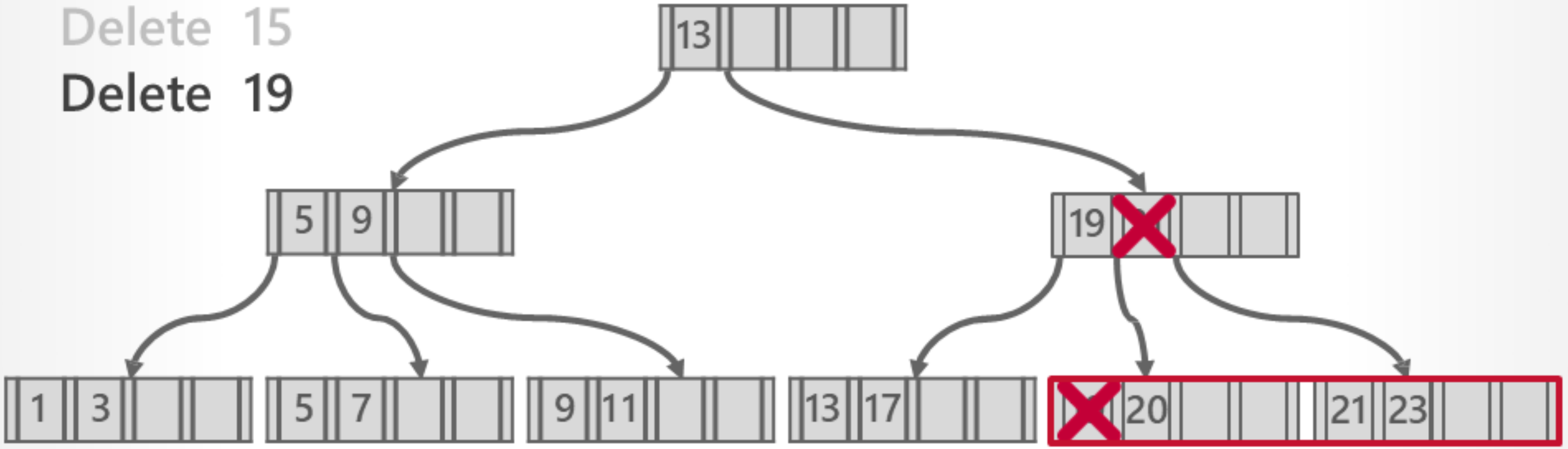
No "rich" sibling nodes to borrow.

Merge with a sibling

B+TREE – DELETE EXAMPLE (3)

Delete 15

Delete 19



Under-filled!

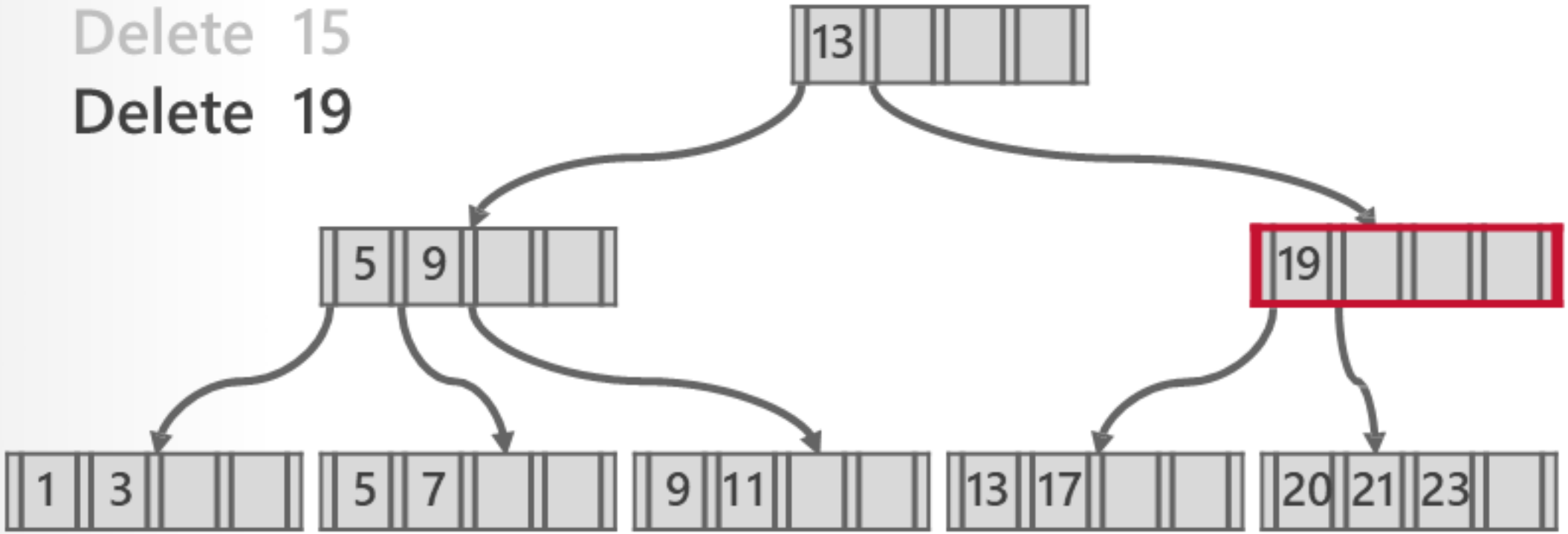
No "rich" sibling nodes to borrow.

Merge with a sibling

B+TREE – DELETE EXAMPLE (3)

Delete 15

Delete 19

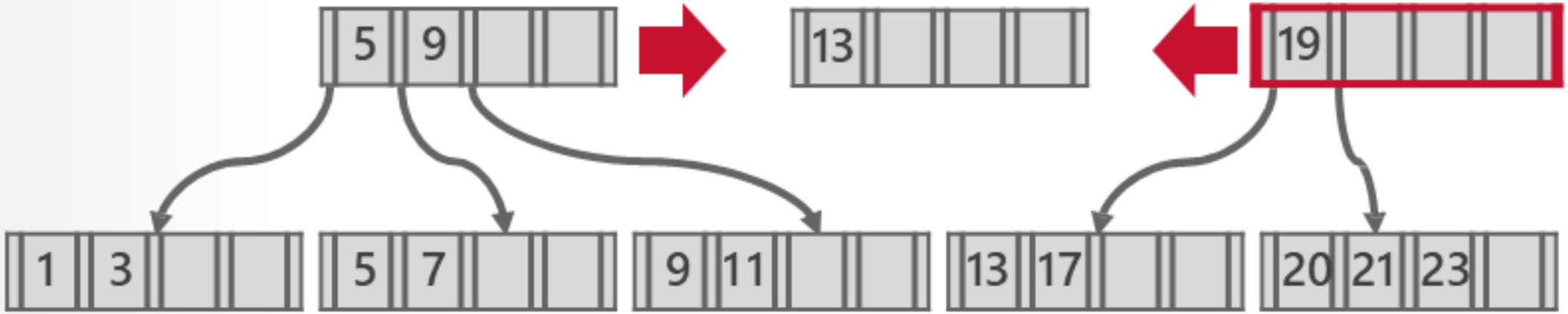


This node is under-filled! Pull-down.

B+TREE – DELETE EXAMPLE (3)

Delete 15

Delete 19

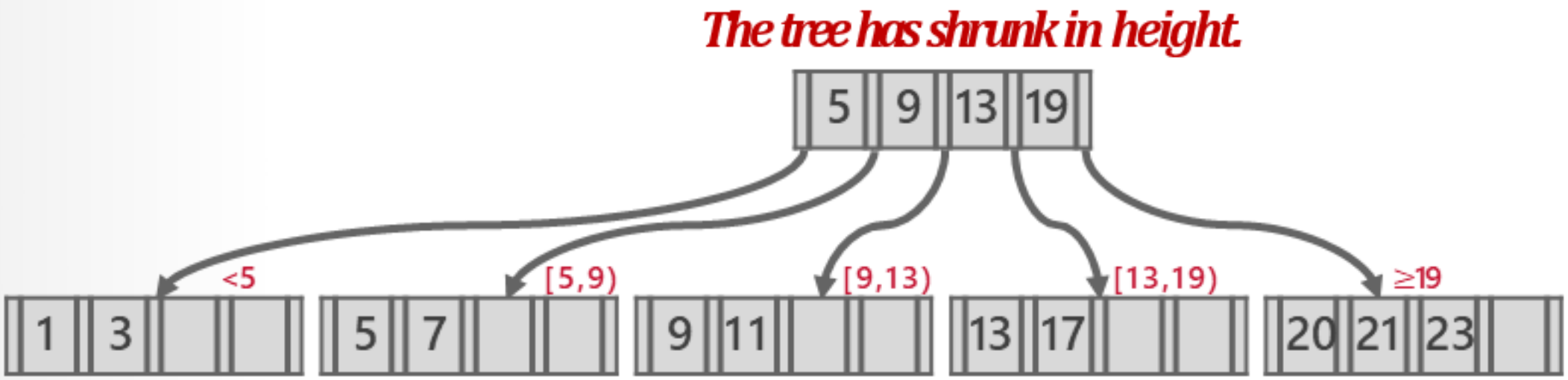


This node is under-filled! Pull-down.

B+TREE – DELETE EXAMPLE (3)

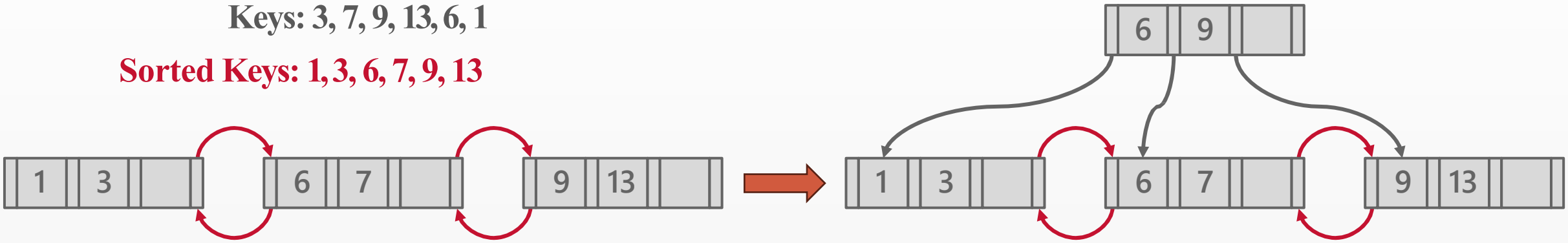
Delete 15

Delete 19

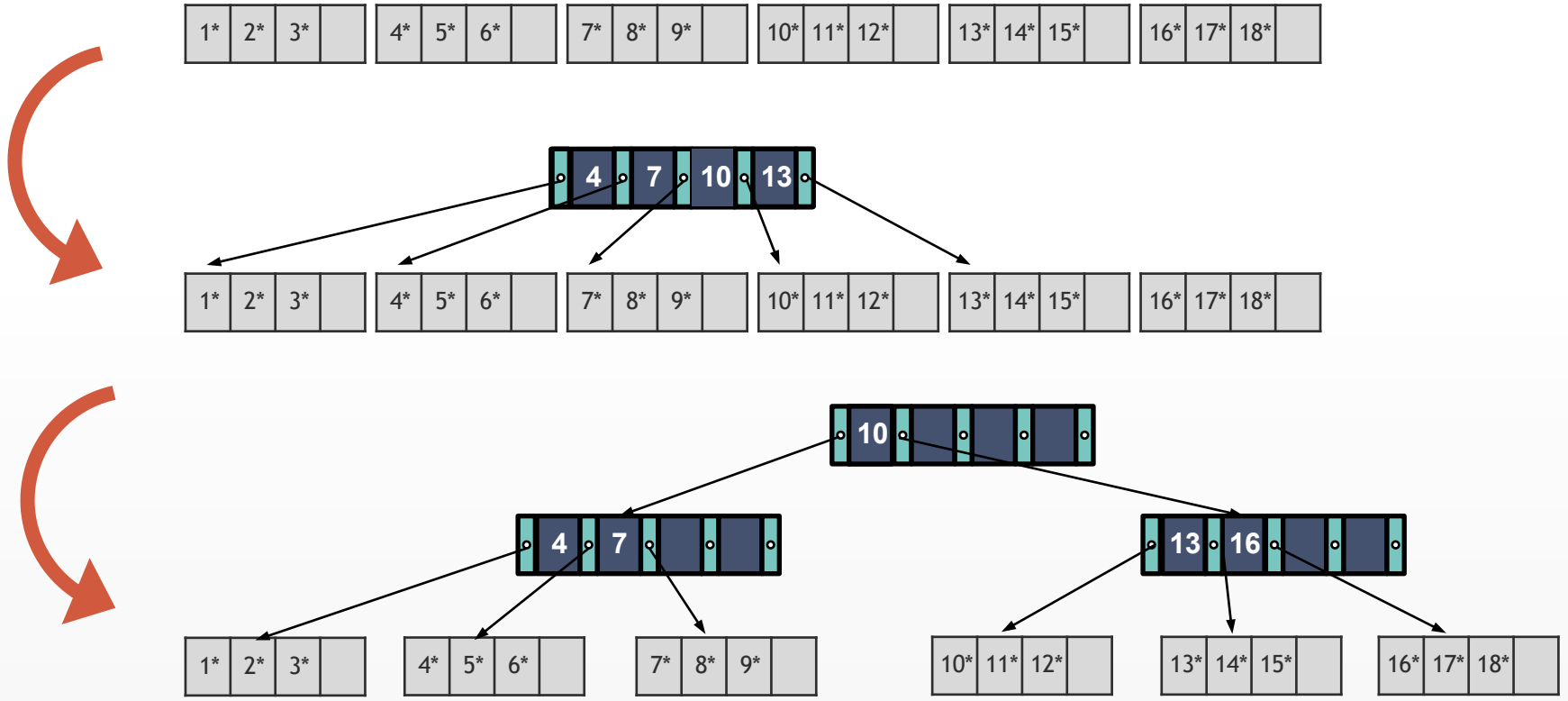


Masovno učitavanje podataka u indeks (*bulk load*)

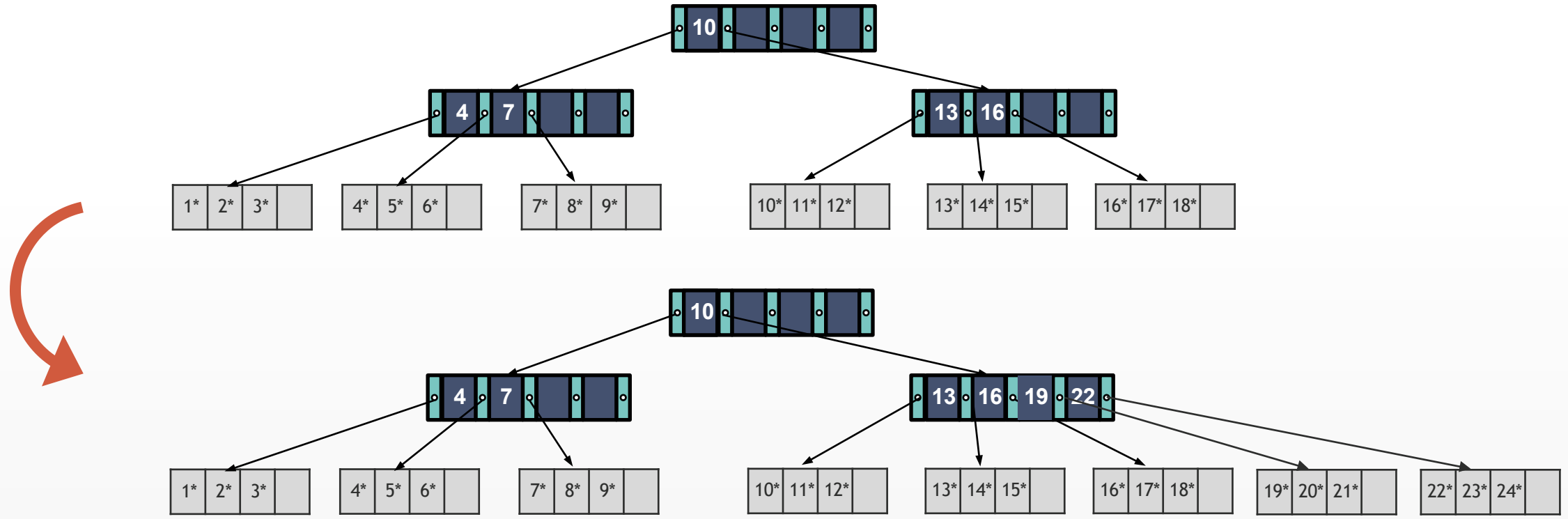
- Dodavanje jednog po jednog ključa je neefikasno.
- Masovno učitavanje je efikasnije ako se stablo gradi odozdo na gore.
  1. Sortiranje ključeva i formiranje povezane liste
     Listovi se ne pune do kraja (obično 2/3)
  2. Dodavanje jednog po jednog roditeljskog čvora (s leva na desno)



Masovno učitavanje podataka u indeks (*bulk load*)



Masovno učitavanje podataka u indeks (*bulk load*)



Kapacitet B+ stabla u praksi

- U praksi se pri formiranju stabla popunjenost čvorova drži na 2/3
  - Neka su strane veličine 138KB, a veličina zapisa (Ključ/Pokazivač) 40KB
  - d je 1600, a popunjenost strane je 2144 parova ključ/pokazivač
- Za ovakav kapacitet jedne strane
  - Stablo visine 1 može da ima 2144^2 parova, dakle oko 4 500 000 ključeva,
  - Stablo visine 2 može da ima 2144^3 parova, dakle oko 9 800 000 000 ključeva

Vizuelizacija dodavanja i brisanja

Simulacija formiranja i pretrage B+stabla

B+ Tree Visualization (usfca.edu)

Crtanje B+ stabla

https://projects.calebevans.me/b-sketcher/

Efikasnost B+ stabla

Uređeni indeksi

Troškovi operacija korišćenjem indeksnog stabla

Pretpostavke

- U listovima indeksa se nalaze reference na slogove u fajlu
- Indeks je klasterišući
- Strane u fajlu su pune 2/3
- Bez duplikata ključa
- Oznake

  B – broj punih strana

  R – broj slogova u bloku

  D – prosečno I/O vreme

  F – (fan-out) izlazni stepen unutrašnjih čvorova

  E – broj ključeva u listu

Operacija		Prosečno vreme
	Heap	Sortiran
Skeniranje	B * D	B * D
Jedna vrednost	B * D / 2	$\log_2 B * D$
Opseg vrednosti	B * D	($\log_2 B$ + strane u opsegu) *D
Dodavanje	2 * D	($\log_2 B$ + B) * D
Brisanje	(B/2+1)*D	($\log_2 B$ + B) * D

Skeniranje

- Indeks nije potreban.
- Pošto je popunjenost 2/3, ukupan broj strana u odnosu na broj kada su strane pune je 3/2B
- Procenjen broj strana

$$\frac{3}{2}B$$

- Procenjeno vreme

$$\frac{3}{2}BD$$

B – broj strana
R – broj slogova u bloku
D – prosečno I/O vreme
F – (fan-out) izlazni stepen unutrašnjih čvorova
E – broj ključeva u listu

Pretraga jedne vrednosti

- Broj strana indeksa koje će biti pročitane = dubina + 1 za koren
- Dodatno jedna strana fajla sa traženim slogom .
- Procenjen broj strana

$$\log_F(BR/E) + 2$$

- Procenjeno vreme

$$(\log_F(BR/E) + 2) * D$$

B – broj strana
R – broj slogova u bloku
D – prosečno I/O vreme
F – (fan-out) izlazni stepen unutrašnjih čvorova
E – broj ključeva u listu

Pretraga opsega vrednosti

B – broj strana
R – broj slogova u bloku
D – prosečno I/O vreme
F – (fan-out) izlazni stepen unutrašnjih čvorova
E – broj ključeva u listu

- Broj strana indeksa koje će biti pročitane
  kada se pronađe prvi list u opsegu čitaju se i susedni dok se ne obuhvate svi listovi u opsegu

$$\log_F(BR/E) + \frac{3}{2}\#listova\_u\_opsegu$$

- Dodatno, strane sa traženim slogovima iz opsega
  (pretpostavljano da je broj listova i strana sa podacima u opsegu isti)

$$\frac{3}{2}\#strana\_u\_opsegu$$

- Procenjen broj strana

$$\log_F(BR/E) + 3 * \#strana\_u\_opsegu$$

- Procenjeno vreme

$$(\log_F(BR/E) + 3 * \#strana\_u\_opsegu) * D$$

Dodavanje i brisanje

B – broj strana
R – broj slogova u bloku
D – prosečno I/O vreme
F – (fan-out) izlazni stepen unutrašnjih čvorova
E – broj ključeva u listu

- Pretpostavka da nema deljenja unutrašnjih čvorova
- Broj strana indeksa i fajla koje se učitavaju

$$\log_F(BR/E) + 2$$

- Dodatno I/O operacije upisa dve ažurirane strane (list i fajl) na disk
- Procenjen broj strana

$$\log_F(BR/E) + 4$$

- Procenjeno vreme

$$(\log_F(BR/E) + 4) * D$$

Uporedna tabela

Operacija	Prosečno vreme		
	Heap	Sortiran	Klasterišući indeks
Skeniranje	B * D	B * D	$\frac{3}{2}BD$
Jedna vrednost	B * D / 2	$\log_2$B * D	$(\log_F(BR/E)+2)*D$
Opseg vrednosti	B * D	($\log_2$B + strane u opsegu) *D	$(\log_F(BR/E)+3*\#strana\_u\_opsegu)*D$
Dodavanje	2 * D	($\log_2$B + B) * D	$(\log_F(BR/E)+4)*D$
Brisanje	(B/2+1)*D	($\log_2$B + B) * D	$(\log_F(BR/E)+4)*D$

Uporedna tabela – asimptotska ocena

Operacija	Prosečno vreme		
	Heap	Sortiran	Klasterišući indeks
Skeniranje	$O(B)$	$O(B)$	$O(B)$
Jedna vrednost	$O(B)$	$O(\log_2 B)$	$O(\log_F B)$
Opseg vrednosti	$O(B)$	$O(\log_2 B)$	$O(\log_F B)$
Dodavanje	$O(1)$	$O(B)$	$O(\log_F B)$
Brisanje	$O(B)$	$O(B)$	$O(\log_F B)$

Uticaj konstanti (#broj strana u opsegu), drugačije brzine čitanja uzastopnih strana sa diska su zanemareni. U praksi imaju uticaja, pa korišćenje indeksa nije uvek bolja opcija