ELTE | IK
INFORMATIKAI KAR

Adatbázisok 2

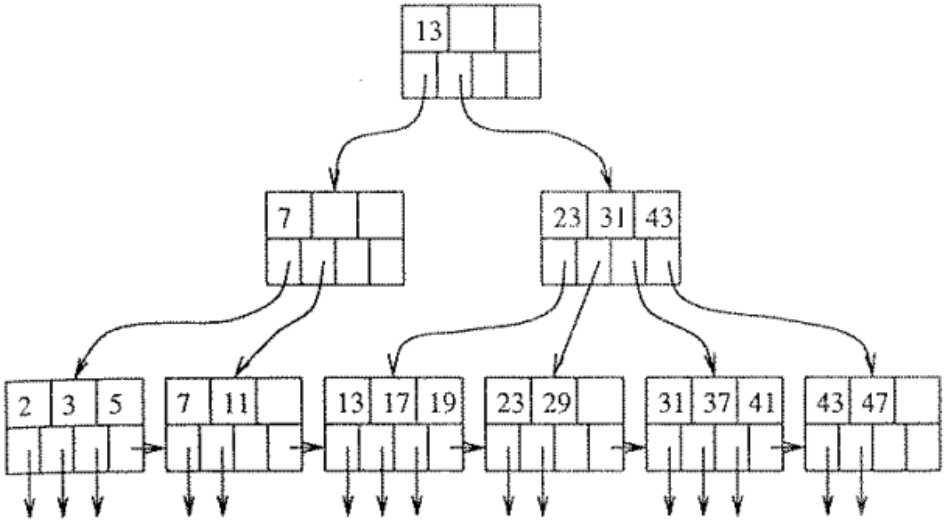
Indexek

B+ fa

- **Kiegyensúlyozott fa**, azaz minden út a gyökértől egy levélig egyforma hosszú.
- Általánosan a B-fák egy adatszerkezetek egy családja. Az évek során sok változat megjelent:
  - B-Tree (1970)
  - B+Tree (1973)
  - B*Tree(1977)
  - B-link Tree (1981)
  - …
- Minden B+ fához tartozik egy paraméter ($\boldsymbol{n}$), amely meghatározza a blokkok elrendezését.
  - Egy blokk legfeljebb $\boldsymbol{n}$ kulcsot és $\boldsymbol{n+1}$ mutatót tartalmazhat.

B+ fa tulajdonságok

- Blokkolvasások száma keresésnél: $t + 1$, ahol t a fa magassága.
- $t \approx log_{n+1}T(R)$ adja meg, ahol az $n + 1$ a mutatók száma.
- A B+ fa alsó szintje sűrű indexnek tekinthető, azaz a levél csúcsok minden kulcsot tartalmaznak.
- A gyökér csúcsban kell lennie legalább 2 használatban lévő mutatónak (kivéve azt a triviális esetet, amikor összesen 1 rekordunk van).

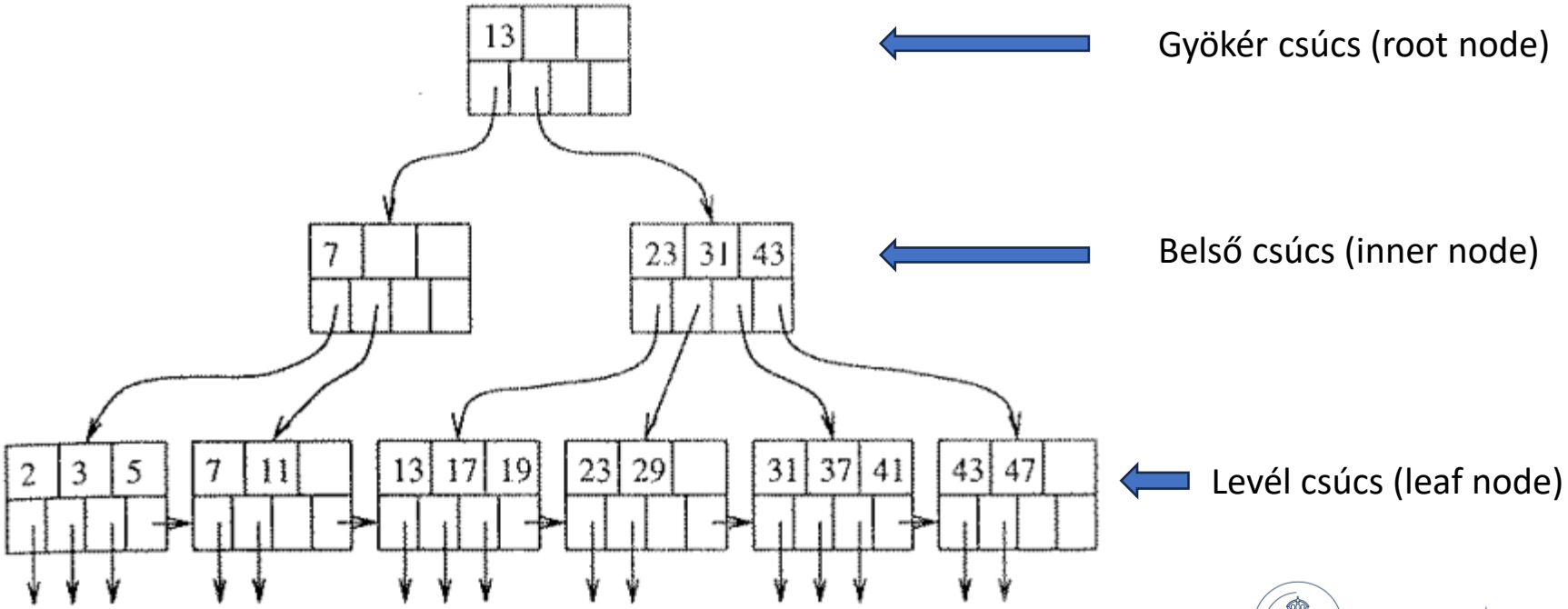


Egyszerű B+ fa (n=3)

B+ fa csúcsai

- A csúcsok kulcs-mutató párokat tartalmaznak:

  <mutató>|<kulcs>|<mutató>|<kulcs>|<mutató>|<kulcs>|<mutató>
- A kulcsok általában rendezve vannak (nem mindig).
- A levél csúcsok mutatói ROWID-k, amelyek a megfelelő blokkokra mutatnak.

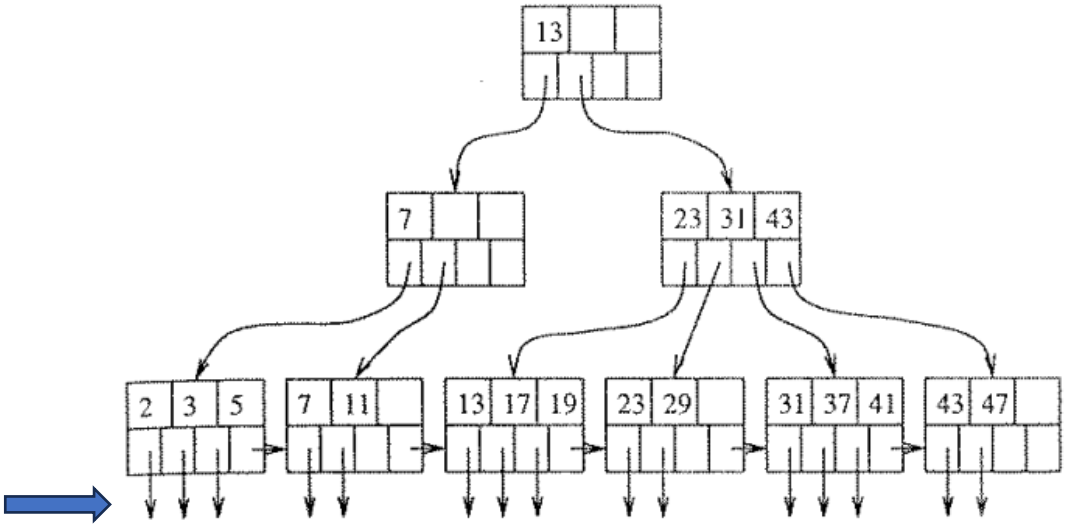


Egyszerű B+ fa (n=3)

B+ fa tulajdonságok (folytatás)

- A levelekben az utolsó mutató a következő levélblokkra mutat.
- Egy levélblokk többi n mutatójából legalább $\lceil (n+1)/2 \rceil$ használatban van és adatrekordra mutat.

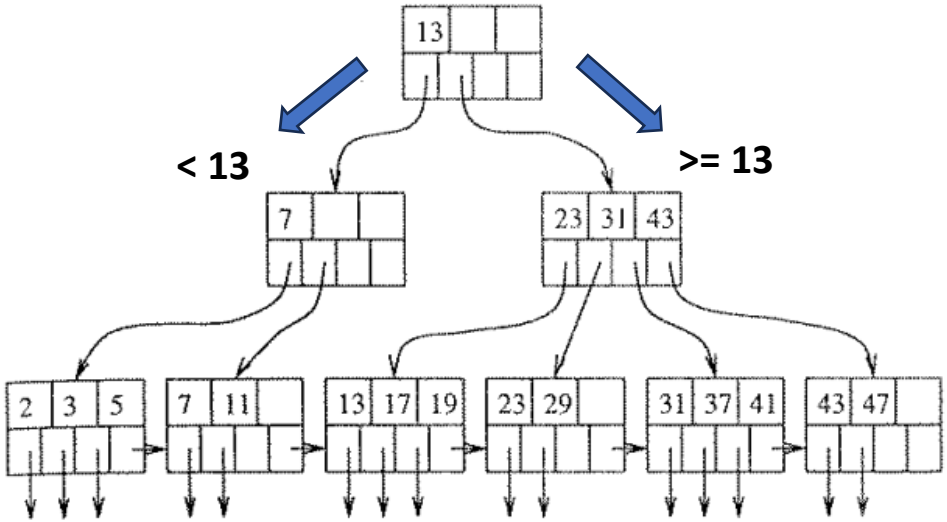
Olyan blokkokra mutatnak, amelyekben rekordok találhatók.



Egyszerű B+ fa (n=3)

B+ fa tulajdonságok (folytatás)

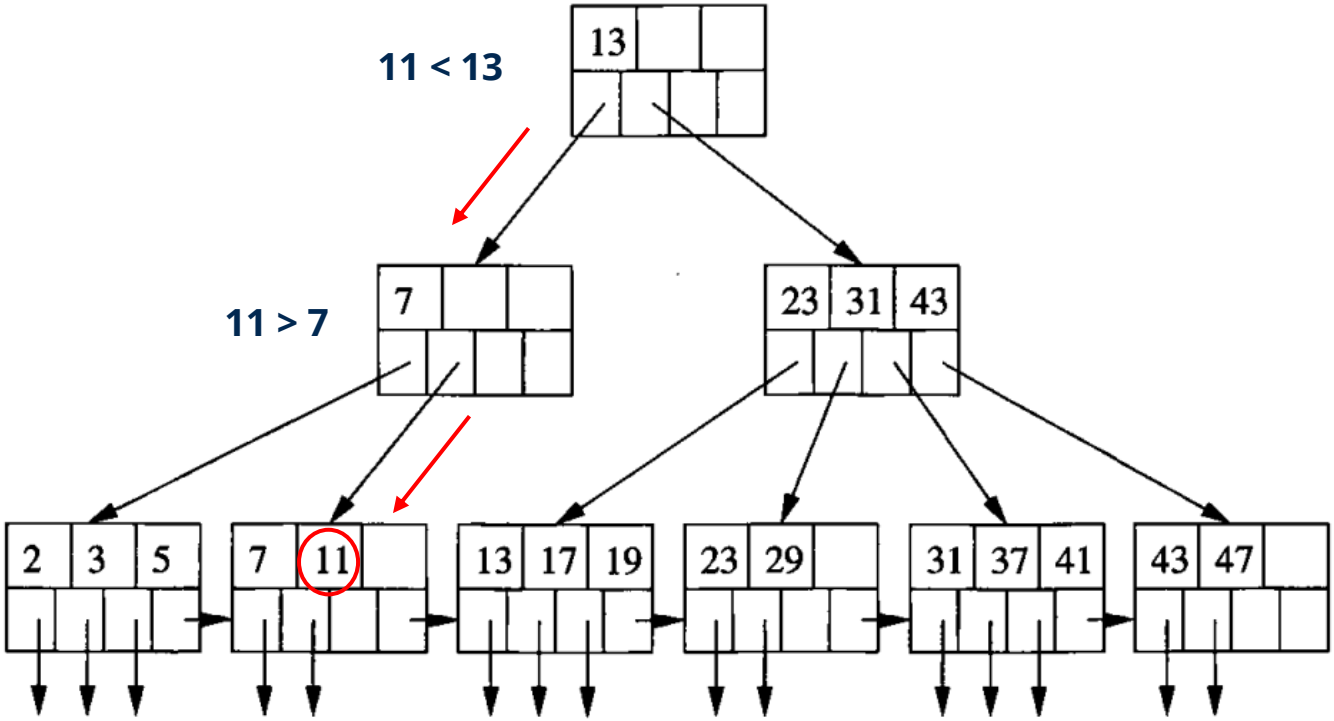
- A közbeeső szinteken lévő csúcsokban mind az $n + 1$ mutató a fa következő szintjére mutat. Közülük legalább $\lceil (n + 1)/2 \rceil$ használatban van.
- Ha $j - 1$ kulcs van ($K_1 K_2 \ldots K_{j-1}$), akkor j mutató van használatban. Az első mutató a fa olyan részére mutat, ahol K_1-től kisebb kulcsok találhatóak
- A második mutató a fa olyan részére mutat, ahol a kulcsok értéke $\geq K_1$ és $\leq K_2$ és így tovább.



Egyszerű B+ fa (n=3)

Keresés B+ fában

- Kereséskor a gyökértől indulunk és levél csúcsok felé haladunk.
- Alkalmas egyenlőség alapú keresésre és tartomány (intervallum) lekérdezéshez is.
- Példa: Keressük meg a 11-es kulcsú rekordot.

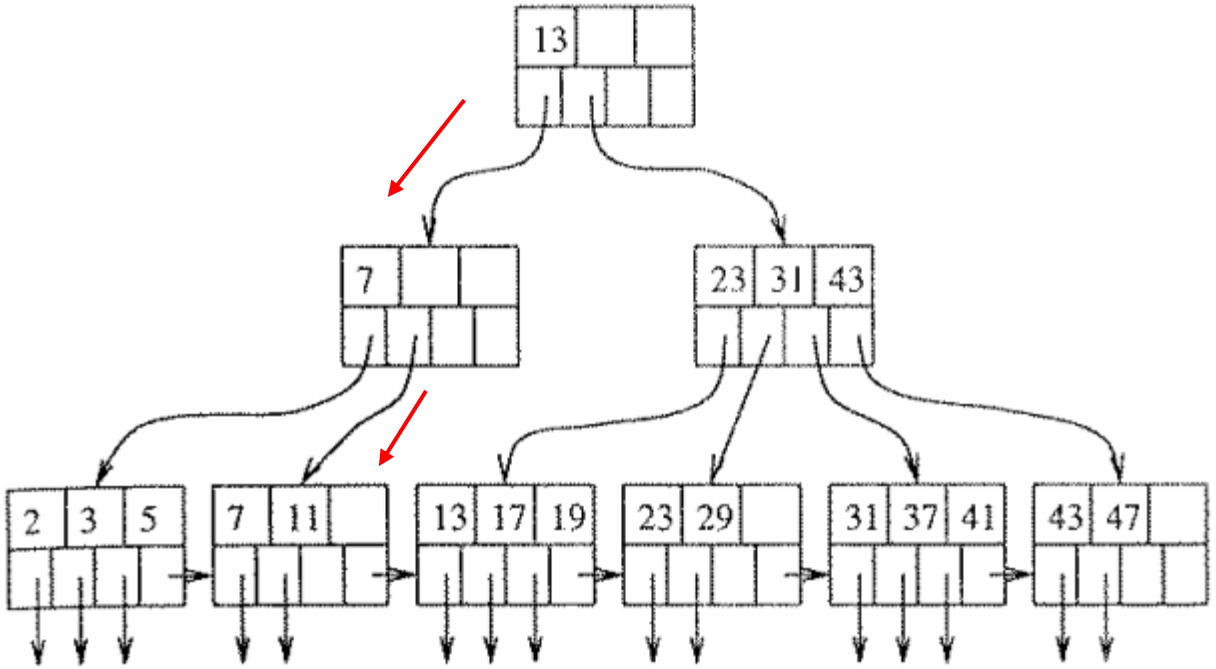


Beszúrás B+ fába

- Keressük meg a megfelelő levél csúcs (mint a keresésnél).
- Ha van üres hely: szúrjuk be az új kulcsot és mutatót.
- Ha nincs üres hely, akkor szét kell vágnunk egy levél csúcsot:
  - Az L levél csúcsban lévő kulcsokat osszuk szét L és egy új L_2 csúcs között.
  - Az L csúcs szülő csúcsába szúrjunk be egy mutatót, amely L_2-re mutat (L_2 legkisebb kulcsát).

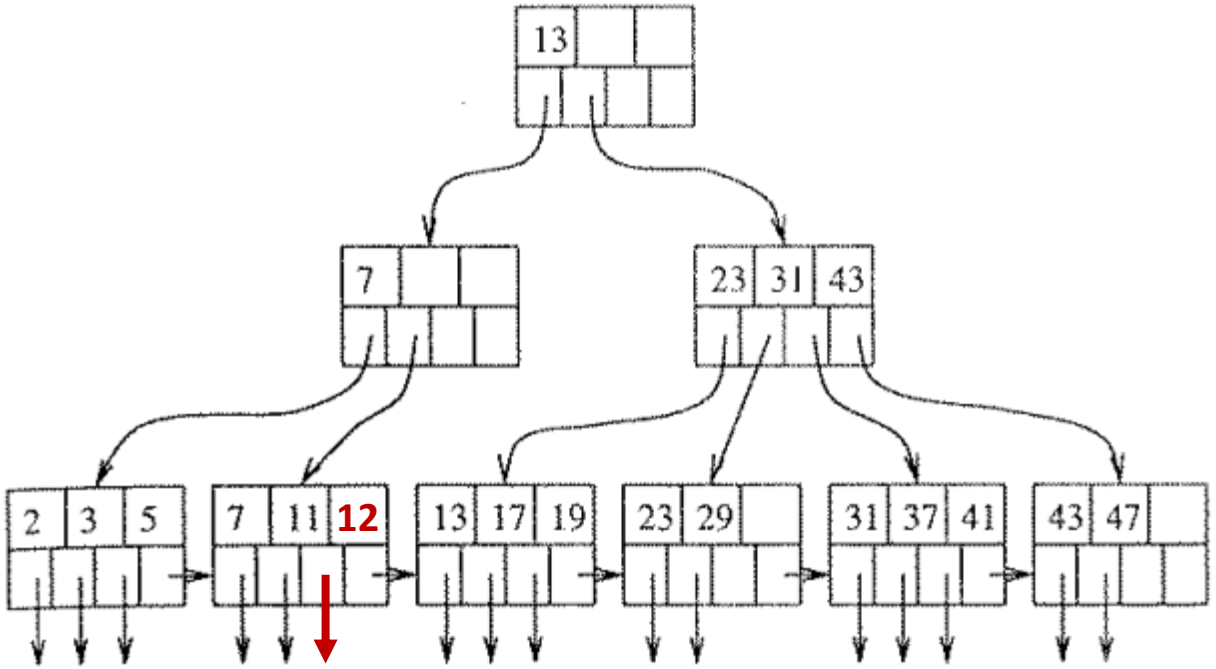
Beszúrás példa: 12

- A 12-es kulcs beszúrása.
  - Keressük meg a megfelelő levél csúcsot.
  - Ha van hely a levélben, szúrjuk be a kulcsot és a mutatót.



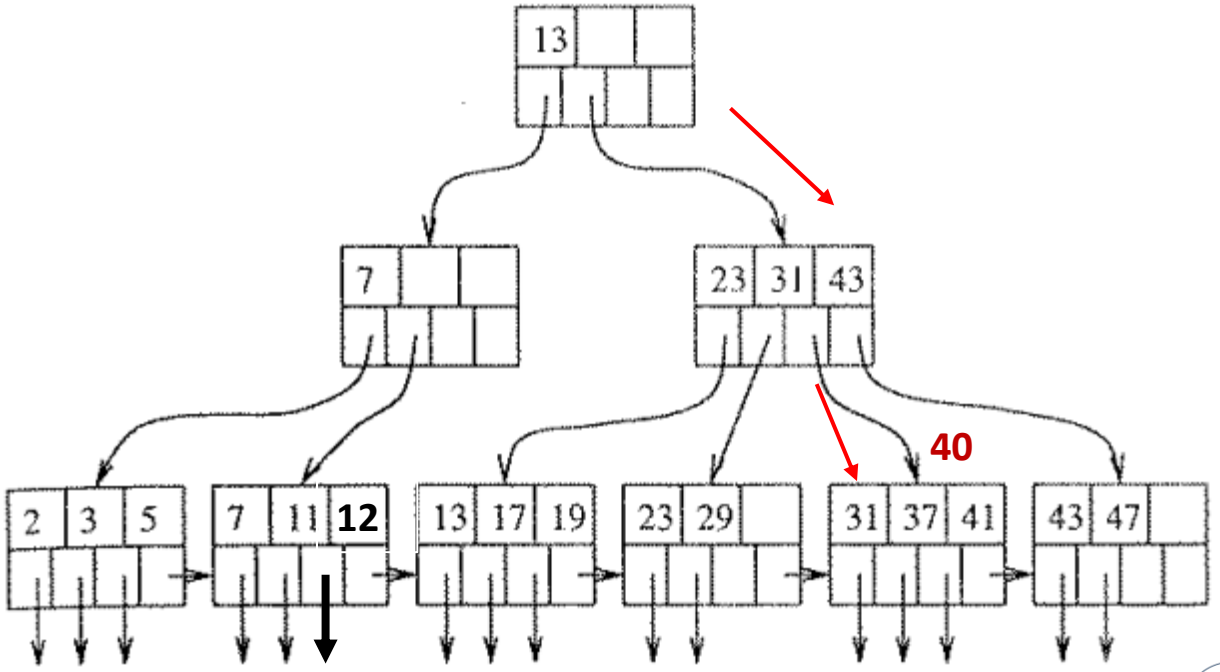
Beszúrás példa: 12

- A 12-es kulcs beszúrása.
  - Keressük meg a megfelelő levél csúcsot.
  - Van hely a levélben, szúrjuk be a kulcsot és a mutatót.



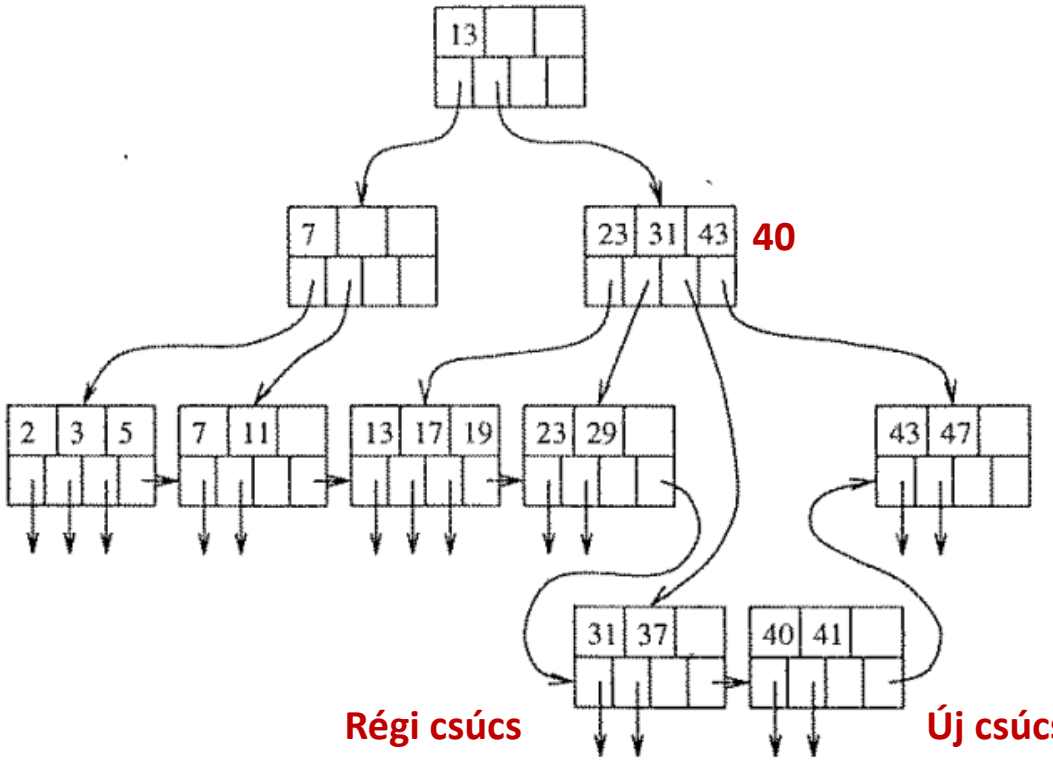
Beszúrás példa: 40

- A 40-es kulcs beszúrása.
  - Keressük meg a megfelelő levél csúcsot.
  - Mivel nincs hely, ezért csúcs szétvágásra van szükség.



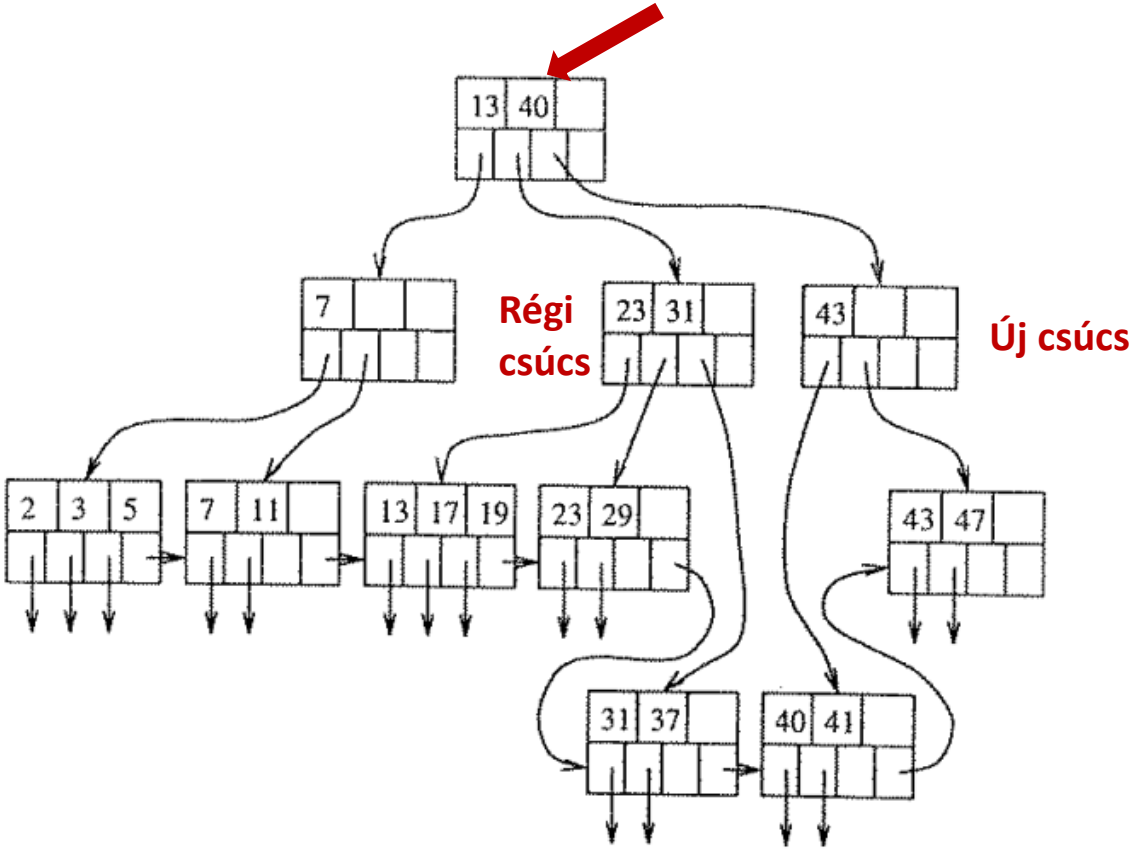
Beszúrás példa: 40

- Létrehozunk egy új csúcsot.
- Szétosztjuk a kulcsokat a régi és új csúcs között.
- Az új csúcsnak kell egy szülő, ezért egy kulcsot fel kell küldenünk (40).



Beszúrás példa: 40

- A szülő csúcsba is be kell szúrnunk egy mutatót, viszont ott sincs hely.
- Ezért a köztes csúcsot is szétvágjuk és a szülő csúcsba beszúrjuk a megfelelő mutatót.

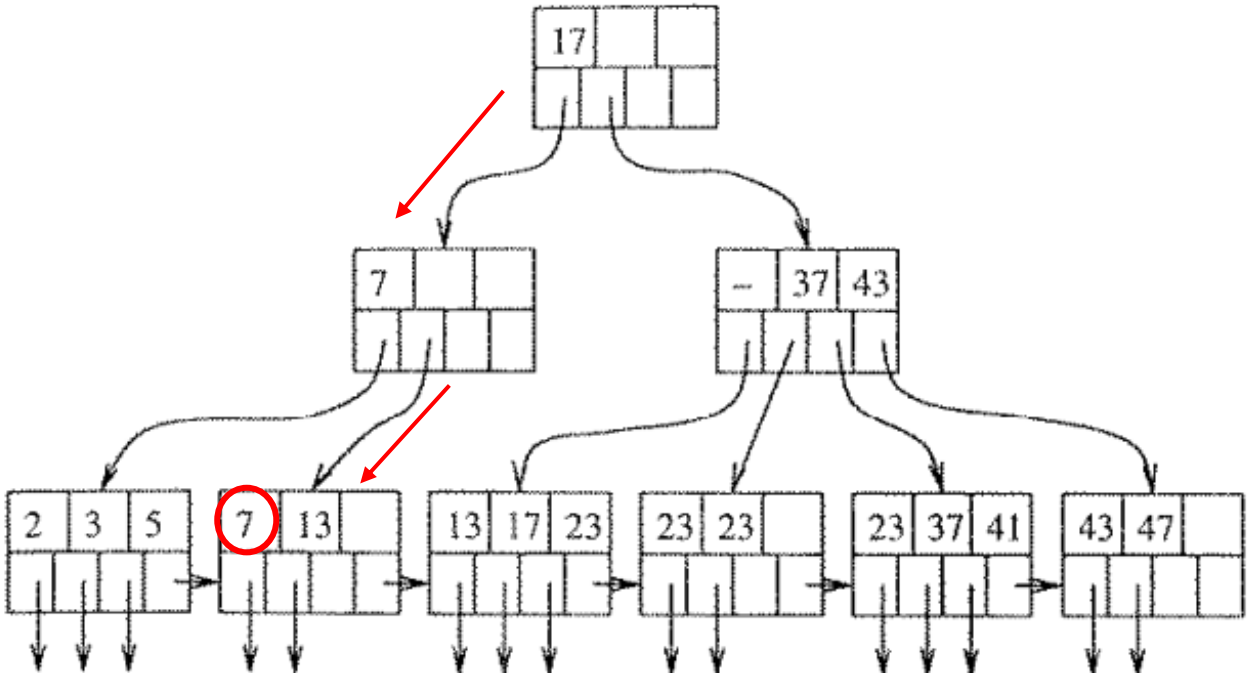


Törlés B+ fában

- A gyökértől indulva keressük meg a rekordot és töröljük az index bejegyzést.
- Ha a csúcs a törlés után is **tartalmaz elegendő kulcsot**, akkor kész is vagyunk!
- Ha törlés után nincs elengedő kulcs két lehetőségünk van:
  - Próbáljunk **kölcsön kérni** kulcsokat a csúcs testvérétől (olyan csúcs, amelynek ugyanaz a szülője). A szülőben lévő kulcsok változhatnak.
  - Ha az átrendezés (kölcsön kérés) nem működik, akkor **össze kell vonni** csúcsokat és törölni a megfelelő mutatókat. Ha a szülőben nincs elengedő kulcs, akkor rekurzívan folytatjuk az összevonást a szülőre.

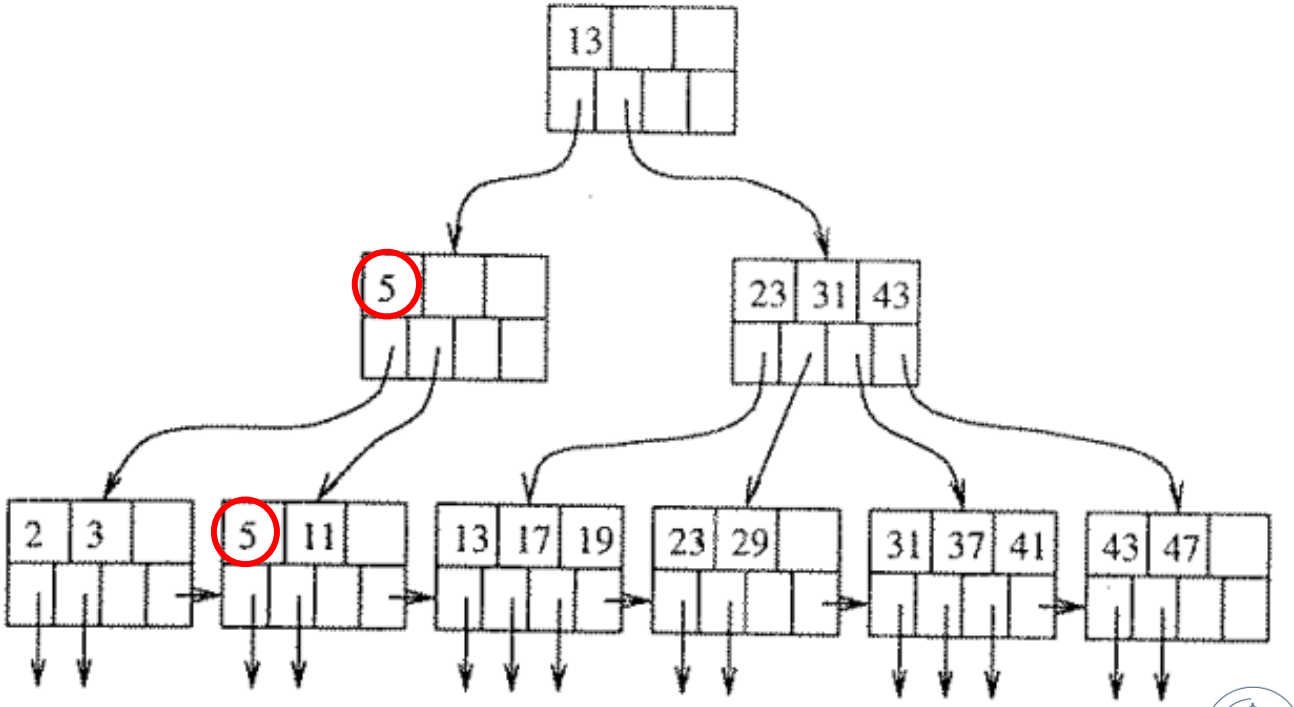
Törlés példa: 7

- Töröljük a 7-es kulcsú rekordot. Először meg kell keresnünk.
- Ha töröljük a rekordot, akkor a levél csúcsban nem marad elegendő rekord, ezért kölcsön kell kérnünk a testvér csúcstól.
- Mivel ott a kölcsön után is lesz elegendő, ezért ezt gond nélkül megtehetjük.



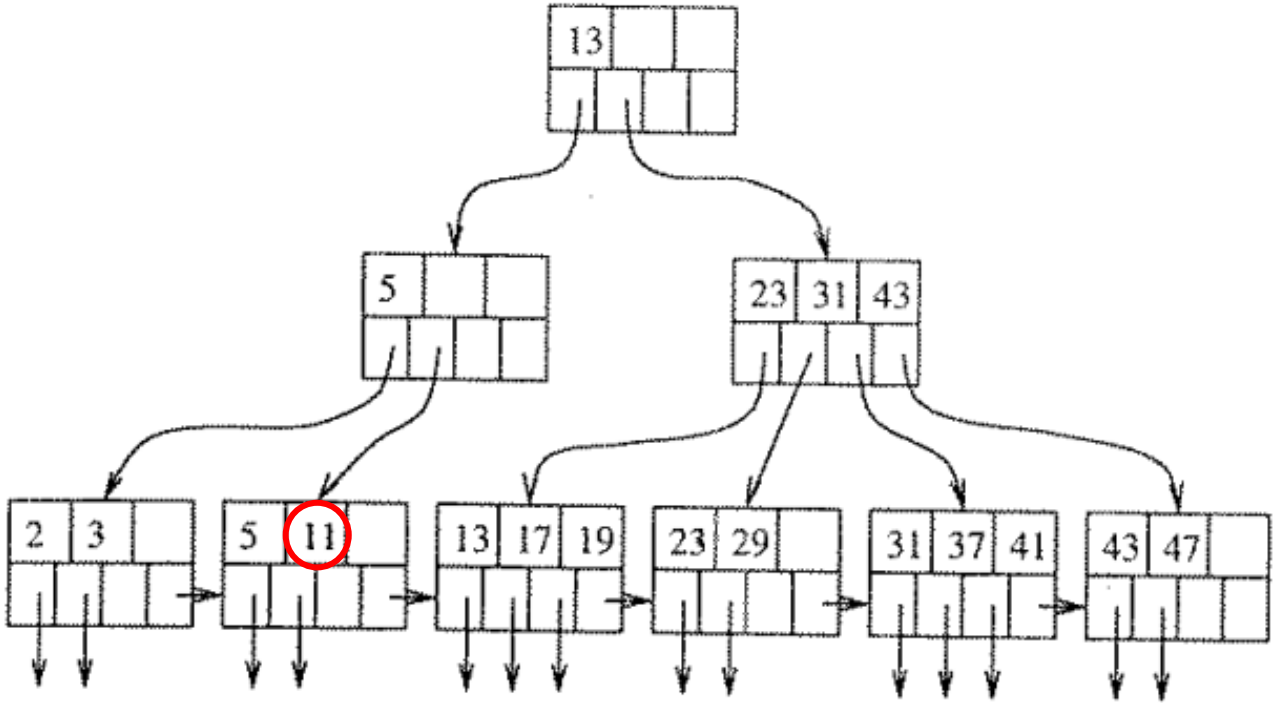
Törlés példa: 7

- Az 5-ös kulcsot kölcsönkértük.
- Ezért a szülőben is megváltoznak a kulcsok. A 7-es helyére az 5-ös kerül, hogy helyes maradjon a B+ fa struktúrája.



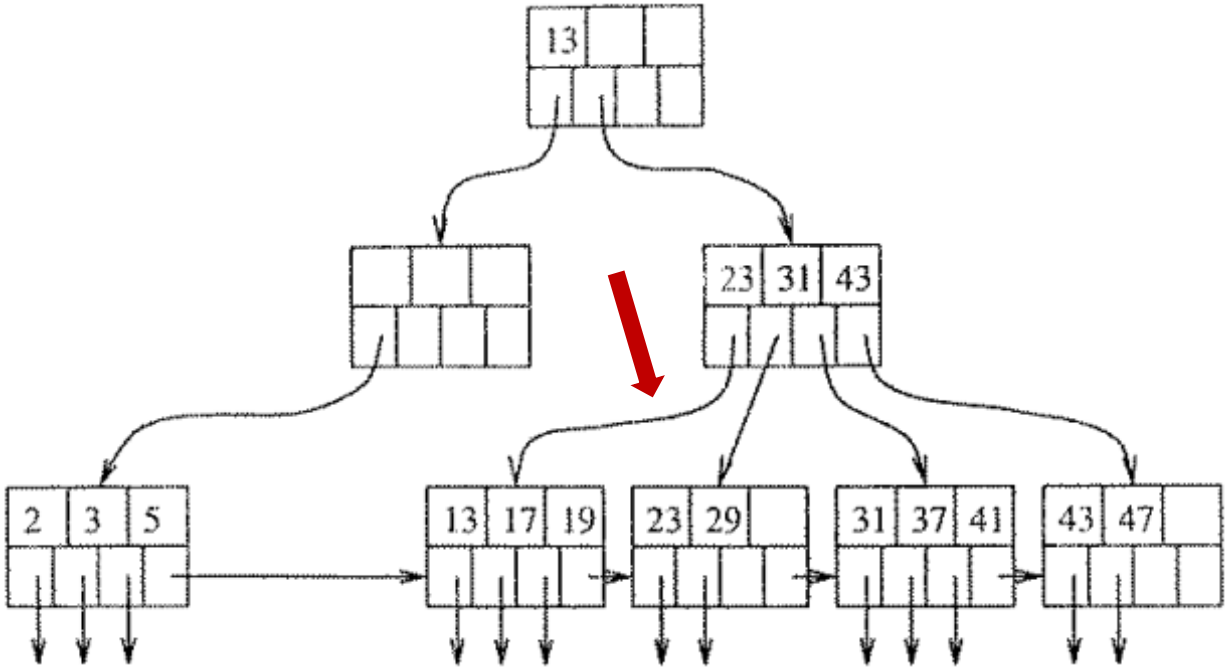
Törlés példa: 11

- Töröljük most a 11-es kulcsú rekordot.
- A törlés után nem marad elengedő kulcs a levélben.
- A csúcsnak egy testvére van, de onnan már nem kérhetünk többet kölcsön, hiszen ott is csak épp elengedő kulcs van.



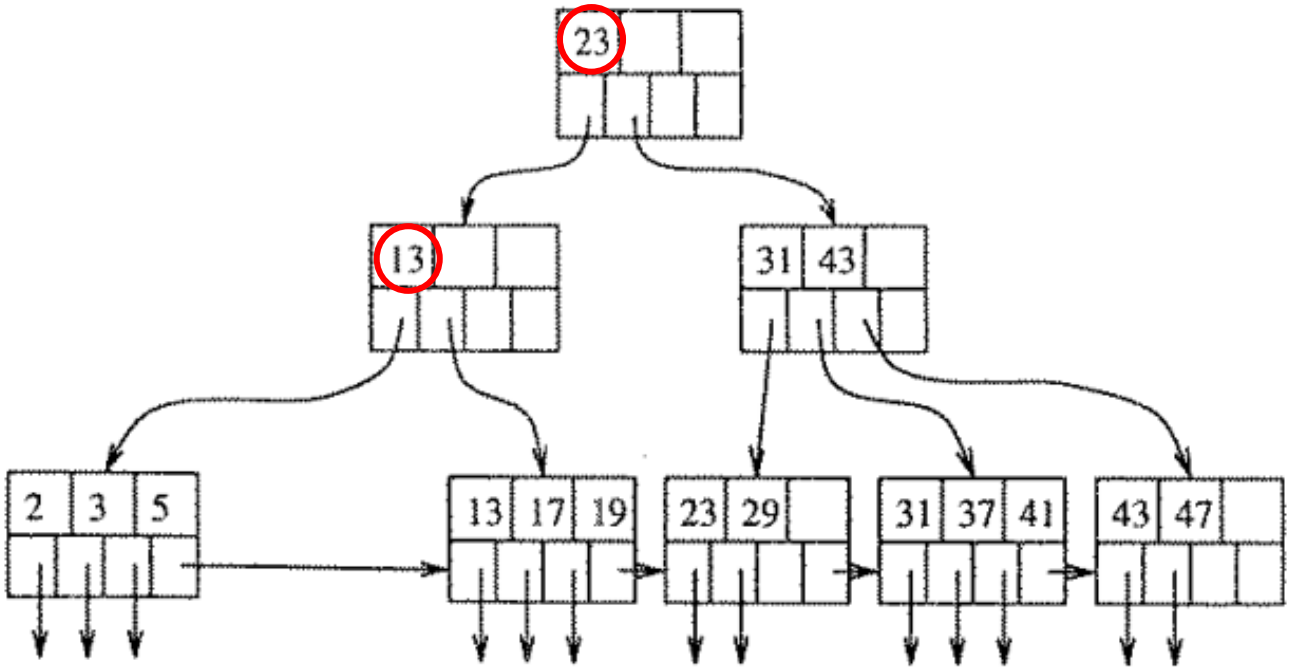
Törlés példa: 11

- A csúcsot, ahol a 11-es volt töröljük a mutatójával együtt.
- Az 5-ös kulcsot áttesszük a testvér csúcsba.
- Így a szülőben nem marad elegendő kulcs.
- Szerencsére a szülő testvér csúcsától kölcsön tudunk kérni.



Törlés példa: 11

- A testvér csúcs átadott egy mutatót, így a bal oldali köztes csúcsba bekerült a 13-as kulcs (ez már elengedő).
- A jobb oldali köztes csúcsból a 23-as kulcs átkerült a gyökér csúcsba.



Törlés B+ fában?

- Valóban szükségünk van törlésre?
- Általában a táblák egyenletesen nőnek (még ha néha törlünk is).
- Egyes rendszerek **késleltetik** a csúcsok összevonását.
    - A késleltetés csökkenti az újraszervezések számát.
- Vannak olyan rendszerek is, ahol hagyják a nem megfelelően telített csúcsok létezését és **periodikusan újraépítik** az egész struktúrát.

Ismétlődő értékek

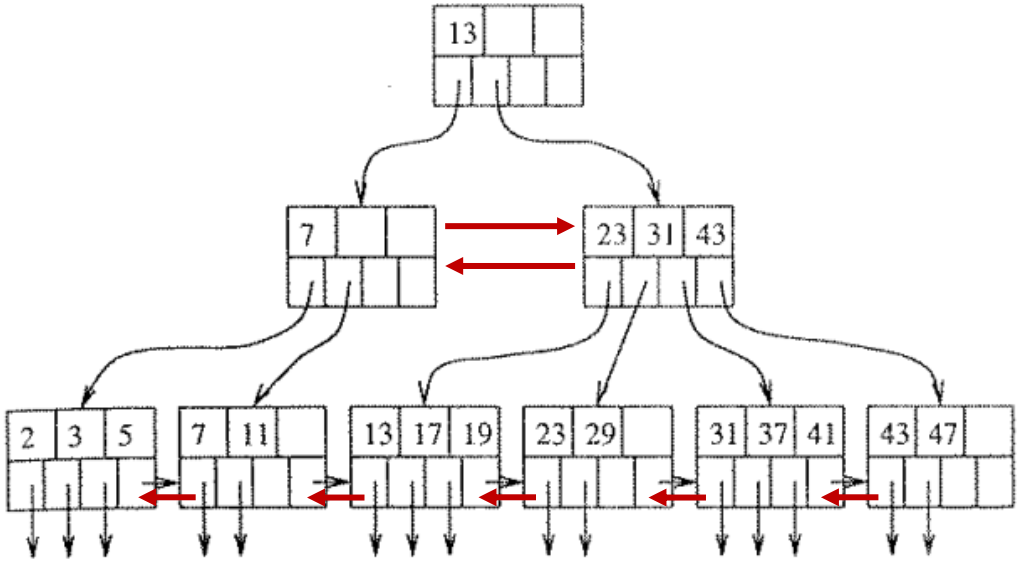
- **Három megközelítés:**
  - Engedjük a NULL kulcsokat (tankönyvben megtalálható).
  - Valamilyen azonosító hozzáfűzése a kulcshoz, hogy egyedi legyen (pl. elsőleges kulcs; blokk azonosító és résjegyzék azonosító).
  - Túlcsordulás blokkok a levél csúcsokhoz.
- Ezek közül a **második** megközelítés a leggyakrabban használt!

B+ fa az adatbázisokban

- Egy csúcs tipikusan egy blokknak felel meg (általában 4-16 KB).
- Így valós esetben egy blokk 500-600 kulcs-mutató párt is tartalmazhat.
- Példa 500 pár esetén:
  - 3 szint esetén: $500 * 500 = 250\ 000$ rekord indexelhető.
  - 4 szint esetén: $500^3 = 125\ 000\ 000$ rekord indexelhető.

B+ fa az adatbázisokban

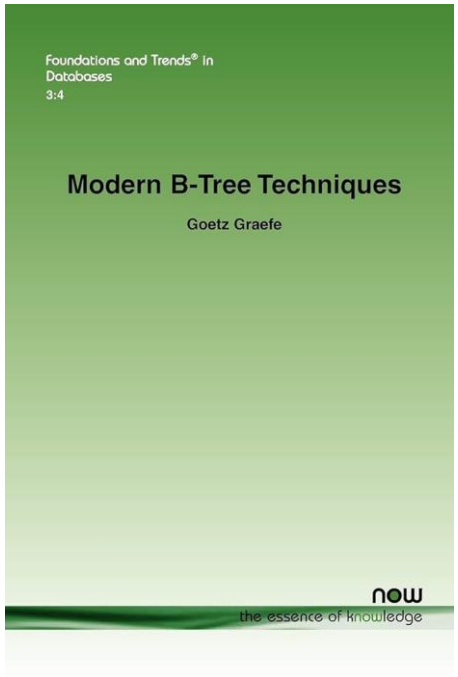
- Általában nem az egyszerű B+ fát implementálják (pl. B-link fa – sibling pointerekkel).
- Egyes rendszerekben a pufferkezelő a gyakran használt indexek gyökér csúcsát mindig az pufferben tartják, így gyakorlatban a szükséges blokkolvasások száma egyel csökken.



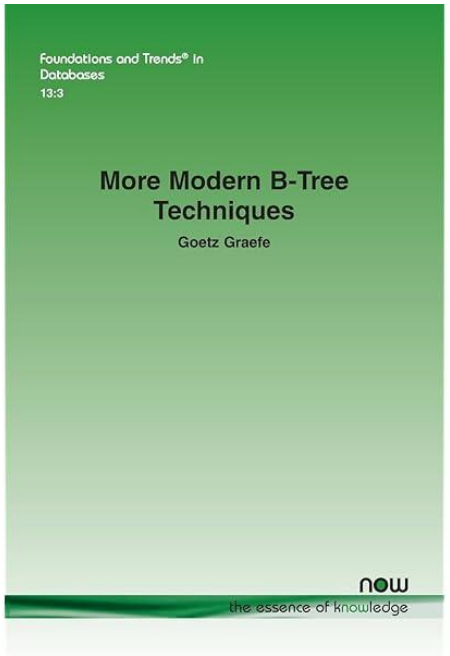
B-link fa

B+ fa az adatbázisokban

- Optimalizálások: deduplication, bluk insert, prefix compression, suffix truncation, fractal trees (delayed update).
- Ajánlott irodalom:



2011



2024

Bittérkép (bitmap) indexek

- Egy oszlopra N hosszú bitvektorokat készítünk, ahol N a sorok száma és annyi bitvektorunk lesz, ahány különböző érték előfordul az oszlopban.

Neptun	Kar	Felvétel éve
ABC123	IK	2018
XYZ789	TTK	2019
ASD135	IK	2020
GOT999	IK	2019

Kar = „IK”	Kar = „TTK”
1	0
0	1
1	0
1	0

Felvétel = 2018	Felvétel = 2019	Felvétel = 2020
1	0	0
0	1	0
0	0	1
0	1	0

Bittérkép indexek

- SELECT * FROM HALLGATOK

  WHERE KAR=„IK” AND FELVÉTEL_ÉVE IN (2018, 2019);

Kar = „IK”
1
0
1
1

AND [

Felvétel = 2018
1
0
0
0

OR

Felvétel = 2019
0
1
0
1

] =

Eredmény
1
0
0
1

Bittérkép indexek a gyakorlatban

- Hasznos, ha:
  - A tábla ritkán módosul (mivel minden módosításnál a bitvektorokat is módosítani kellene)
  - Az oszlopok kardinalitása alacsony.
  - Egyenlőség alapú lekérdezéseknél.
- A népszerű relációs rendszerek közül az Oracle és IBM Db2 implementálja.
- Főleg adattárházaknál van használatban (ritkán módosul).
- Újabb rendszerekben is megtalálható valamilyen formában (pl. Miluvs).

További indexstruktúrák

- Térbeli indexek (R-fa, kd-tree, octree) -> MSc (Információs Rendszerek szakirány)
- Vektor adatbázisokban (legközelebbi szomszéd keresésekhez): Inverted File, Hierarchical Navigable Small World
- Tartalmazás vizsgálat: bloom filter
- Új trend: **learned index**

Tankönyv fejezetek

- Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom:

  Adatbázisrendszerek megvalósítása

  - 4. fejezet: Indexstruktúrák (4.3, 5.4 fejezetek)
- Silberschatz, Korth, & Sudarshan: **Database System Concepts**
  - Chapter 14. Indexing (14.3, 14.4, 14.9)