Algoritmusok és adatszerkezetek II. 3. Előadás

B+ fa fogalma, láncolt és szöveges ábrázolása, elhelyezkedése a háttértáron, magassága, műveletei. Általános fák.

Tartalom

B+ fák

- Nagy mennyiségű adatok tárolására kiválóan használható adatszerkezet
- Többszintű hierarchikus indexstruktúra
  - a listákból nem közvetlenül rekordokra, hanem újabb indexlistákra történik hivatkozás
  - minden közvetlenül a rekordokra mutató listája a fa azonos szintjén helyezkedik el
    - Minden rekordot közel azonos idő elérni
- Az indexszerkezet kiegyensúlyozott
  - az indexlisták kihasználtsága is jó
    - minden lista a legelsőt kivéve a kapacitásának minimum a felét kihasználja

Miért jó a B+ fa?

- Rendezett tárolás
  - Nagy adathalmazokon hatékonyabb így a keresés és módosítás
- Szekvenciális elhelyezés – nem praktikus
  - A legtöbb beszúrás és törlés kapcsán a tároló jelentős részének újraírása válna szükségessé
- Lista használata
  - Keresés nem hatékony

Miért jó a B+ fa?

- Keresőfa (pl. AVL fák, vagy piros-fekete fák)
  - Hátrány: Ha az adatokat egy véletlen elérésű háttértáron, pl. egy mágneslemezen kívánjuk elhelyezni.
    - A mágneslemezek: egyszerre az adatok egy egész blokkját (512 byte vagy 4 KB) adatot mozgatja
    - Egy bináris keresőfa egy csúcsa ennek csak egy töredékét használná.
- B+ fa:
  - Jobban kihasználja a mágneslemez blokkjait

A fokszám megválasztása

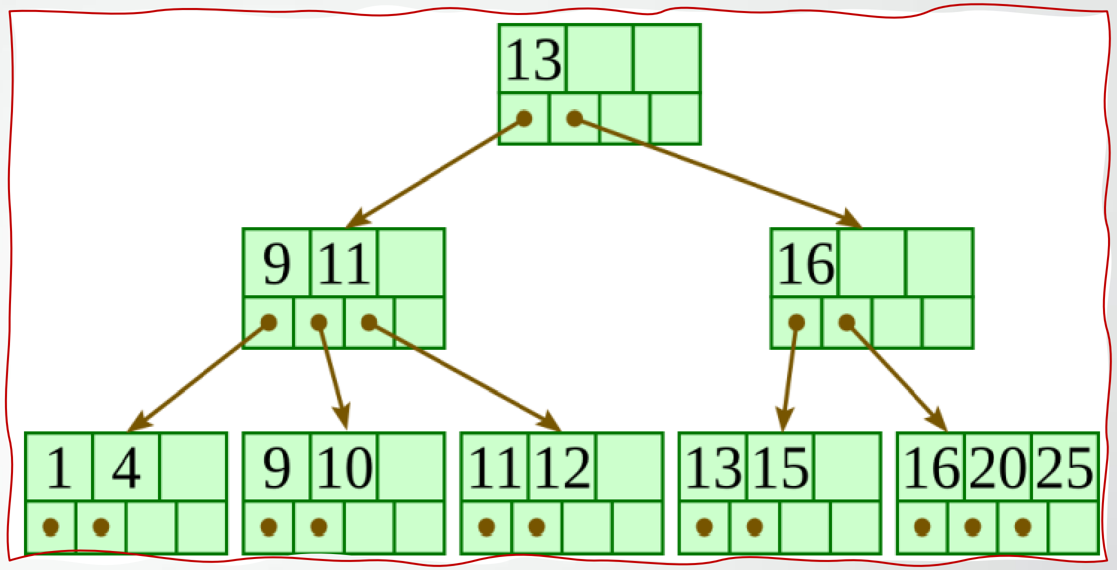
- Egy *d*-edfokú B+ fa csúcsaiban 4 bájtos kulcsok és 6 bájtos pointerek vannak. A B+ fát mágneslemezen tároljuk, ahol a blokkméret 4096 bájt. Mekkorának érdemes választani a B+ fa *d* fokszámát?
  - Egy csúcs legfeljebb d-1 kulcsot és d mutatót tartalmaz
    - $4(d-1) + 6d \leq 4096$

      $10d - 4 \leq 4096$

      $10d \leq 4100$

- d =410 –nek érdemes választani a B+ fa fokszámát

B+ fák



- Minden csúcs
  - legfeljebb *d* mutató (4 ≤ *d*)
  - legfeljebb *d*-1 kulcs
  - ahol *d* a B+ fa fokszáma
    - fára jellemző állandó
- A belső csúcsok: hasító kulcsok (split key)
  - mindegyik referencia két kulcs "között" van:
    - egy olyan részfa gyökerére mutat, amiben minden érték a két kulcs között található
    - (mindegyik csúcshoz hozzáképzelve balról egy "mínusz végtelen", jobbról egy "plusz végtelen" értékű kulcsot)
    - $k_i \leq k < k_{i+1}$
- Az adatok a levélszinten vannak
  - A levélszinten minden kulcshoz tartozik egy mutató, ami a megfelelő adatrekordra hivatkozik. (A leveleket a *d*-edik mutatókkal gyakran listába fűzik.)
  - A gyökércsúcstól mindegyik levél azonos távolságra van

B+ fa műveletidő

- Keresés:
  - A belső csúcsokban található *hasító kulcsok* segítségével
  - A csúcsokban is logaritmikusan keresve, és mindig a megfelelő ágon tovább haladva
  - O(lg *n*) lépésben (ahol *n* a B+ fával ábrázolt adathalmaz mérete)
- A hasító kulcsok nem feltétlenül szerepelnek a levelekben.
- A fa magassága: O(lg *n*)
  - A gyakorlatban a fa *h* magassága az lg *n* érték töredéke:
    - $\log[d](n/(d-1)) \leq h \leq \log[\lfloor d/2 \rfloor](n/2)$

B+ fa műveletidő a gyakorlatban

- A gyakorlatban: a fa *h* magassága az lg *n* érték töredéke:
  - $\log_{d}(n/(d-1)) \leq h \leq \log_{\lfloor d/2 \rfloor}(n/2)$
- Példa:
  - *d*=410 értékkel:
    - $\log_{410}(n/409) \leq h \leq \log_{205}(n/2)$
  - *n*=1.000.000.000 -> 2,4 < *h* < 3,8 -> *h*=3 -> a fának pontosan négy szintje van
  - ➢ Egy-egy adat eléréséhez összesen maximum 3-szor olvasunk a lemezről
    - A felső két szintet az adatbázis megnyitásakor betöltjük a központi tárba
    - A levélszintről még egyet lépünk a tényleges adatrekord eléréséhez.
    - Ha a keresett kulcsú rekord nincs az adatbázisban: csak kétszer olvasunk a lemezről.

d-ed fokú B+ fa invariánsai ($4 \le d$ állandó)

- Minden levélben legfeljebb d-1 kulcs, és ugyanennyi, a megfelelő (azaz ilyen kulcsú) adatrekordra hivatkozó mutató található.
- A gyökértől mindegyik levél ugyanolyan távol található.
  (Más szavakkal, minden levél azonos mélységben, a legalsó szinten van.)
- Minden belső csúcsban eggyel több mutató van, mint kulcs, ahol d a felső határ a mutatók számára.
- Minden *Cs* belső csúcsra, ahol k a *Cs* csúcsban a kulcsok száma: az első gyerekhez tartozó részfában minden kulcs kisebb, mint a *Cs* első kulcsa; az utolsó gyerekhez tartozó részfában minden kulcs nagyobb-egyenlő, mint a *Cs* utolsó kulcsa; és az i-edik gyerekhez tartozó részfában ($2 \le i \le k$) lévő tetszőleges r kulcsra $Cs.kulcs[i-1] \le r < Cs.kulcs[i]$.

d-ed fokú B+ fa invariánsai
($4 \leq d$ állandó)

- A gyökércsúcsnak legalább két gyereke van
(kivéve, ha ez a fa egyetlen csúcsa, következésképpen az egyetlen levele is).
- Minden, a gyökértől különböző belső csúcsnak legalább $\lfloor d/2 \rfloor$ gyereke van.
- Minden levél legalább $\lfloor d/2 \rfloor$ kulcsot tartalmaz (kivéve, ha a fának egyetlen csúcsa van).
- A B+ fa által reprezentált adathalmaz minden kulcsa megjelenik valamelyik levélben, balról jobbra szigorúan monoton növekvő sorrendben.

B+ fa beszúrás

- Ha a fa üres
  - új levélcsúcs létrehozása (gyökércsúcs)
  - beszúrjuk a kulcs/mutató párt
- Különben keressük meg a kulcsnak megfelelő levelet
- Ha a levélben már szerepel a kulcs,
  - a beszúrás sikertelen.
- Különben menjünk az 1. pontra!

1. Ha a csúcsban van üres hely
   - szúrjuk be a megfelelő kulcs/mutató párt kulcs szerint rendezetten ebbe a csúcsba!

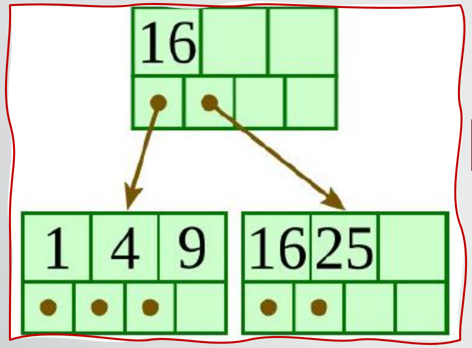
B+ fa beszúrás

2. Ha a csúcs már tele van
   - vágjuk szét két csúccsá
   - osszuk el a *d* darab kulcsot egyenlően a két csúcs között!
     - Ha a csúcs egy levél,
       - vegyük a második csúcs legkisebb értékének másolatát
       - és ismételjük meg ezt a beszúró algoritmust, hogy beszúrjuk azt a szülő csúcsba!
     - Ha a csúcs nem levél,
       - vegyük ki a középső értéket a kulcsok elosztása során,
       - és ismételjük meg ezt a beszúró algoritmust, hogy beszúrjuk ezt a középső értéket a szülő csúcsba!
     - (Ha kell, a szülő csúcsot előbb létrehozzuk. Ekkor a B+ fa magassága nő.)

B+ fa műveletek:

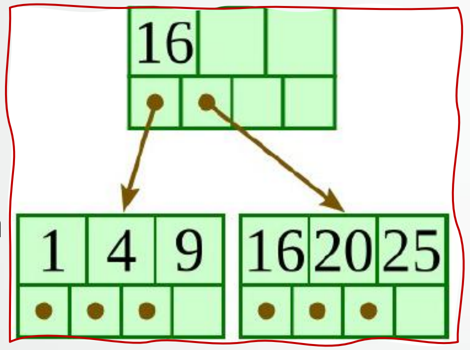
- Beszúrás, törlés
- Konkrét algoritmust nem tárgyalunk
- A példákban általában $d=4$ értékkel dolgozunk
  - A fenti invariánsok alapján:
  - Minden levél legalább két kulcsot tartalmaz
  - Minden belső csúcsnak legalább két gyereke és legalább egy hasító kulcsa van

B+ fa beszúrás példa



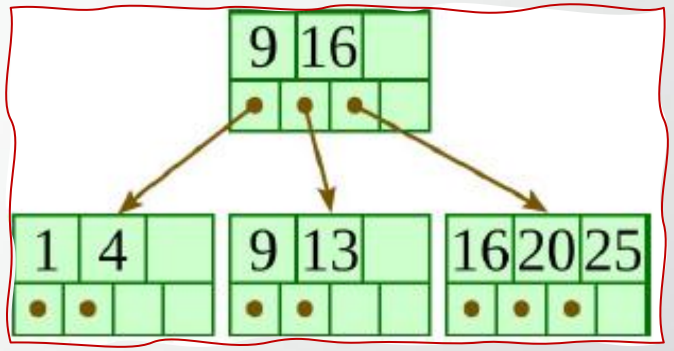
Ins (20)

Beszúródik a megfelelő levélbe.



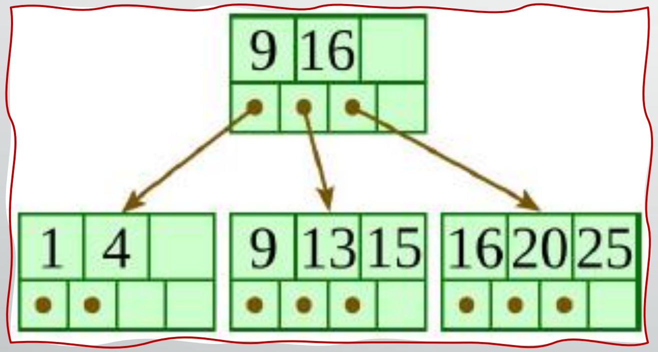
Ins (13)

(1; 4 | 9; 13) hasad a 2. levél min. (9) a szülőbe másolódik új hasítókulcsnak



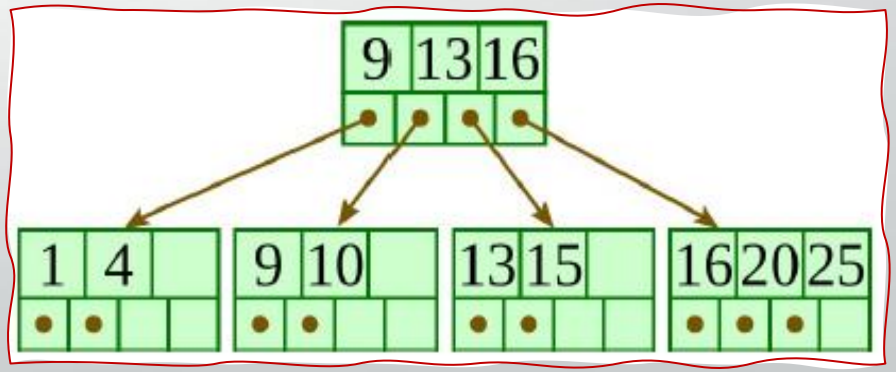
Ins (15)

Beszúródik a megfelelő levélbe.

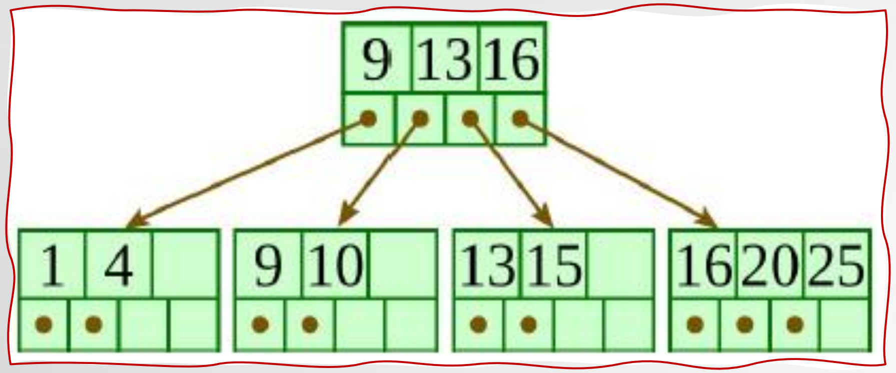


Ins (10)

(9; 10 | 13; 15) hasad a min(13; 15) a szülőbe másolódik új hasítókulcsnak

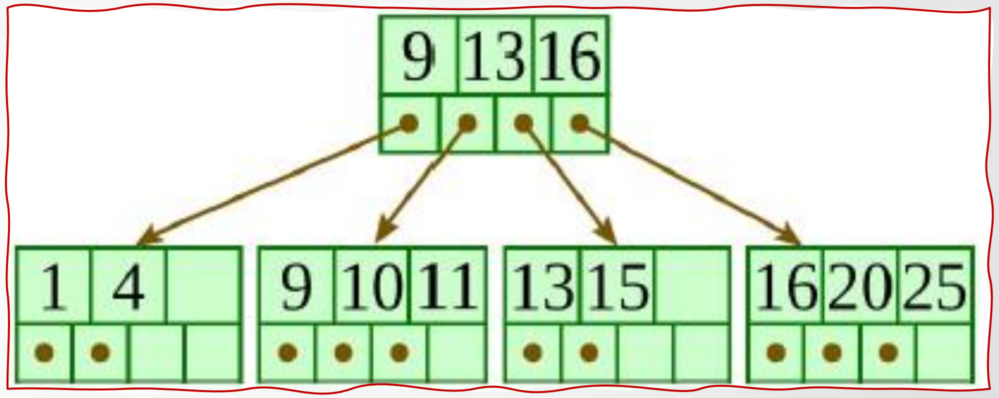


B+ fa beszúrás példa



Ins (11)

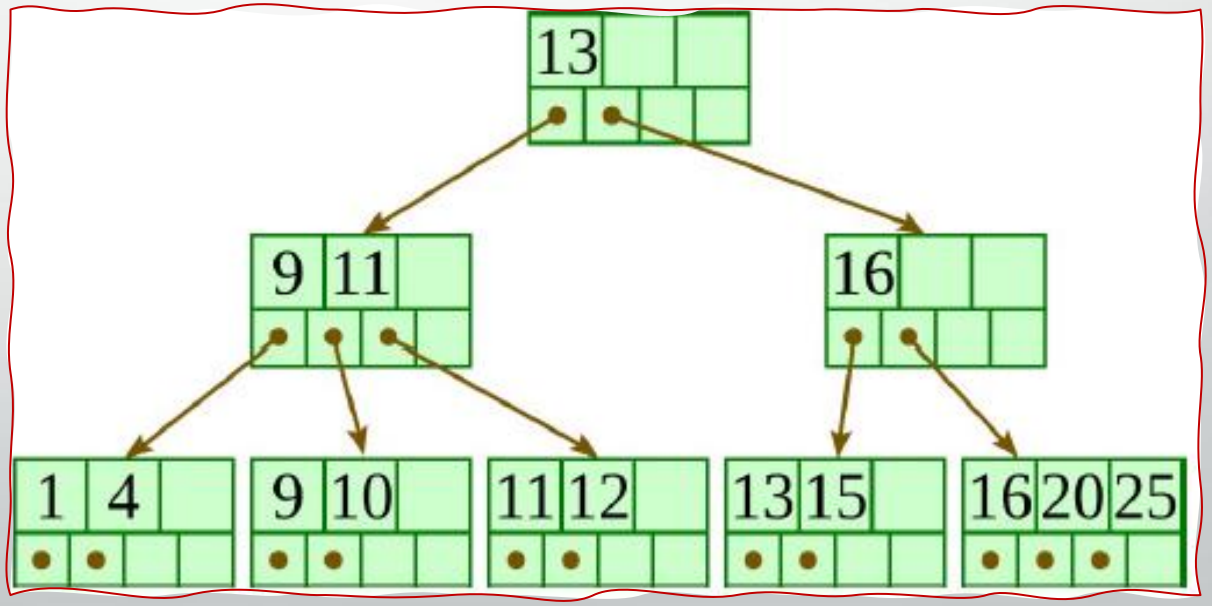
Beszúródik a megfelelő levélbe.



Ins (12)

(9; 10 | 11; 12) hasad
a min(11; 12) a szülőbe másolódik új hasítókulcsnak

(9; 11 | 13; 16) hasad
a min(13; 16) felmegy az újonnan létrehozott gyökérbe



B+ fa törlés

- Keressük meg a törlendő kulcsot tartalmazó levelet.
  - Ha ilyen nincs, a törlés meghiúsul
- Különben
  - a törlő algoritmus futása
    - vagy az A esettel fejeződik be;
    - vagy a B esettel folytatódik,
    - ami után a C eset (nullaszor, egyszer, vagy többször) ismétlődhet,
    - és még a D eset is sorra kerülhet végül.

B+ fa törlés: A és B eset

A. A keresés során megtalált levélcsúcs egyben a gyökércsúcs is:
- Töröljük a megfelelő kulcsot és a hozzá tartozó mutatót a csúcsból.
- Ha a csúcs tartalmaz még kulcsot
  - kész vagyunk.
- Különben töröljük a fa egyetlen csúcsát és üres fát kapunk.

B. A keresés során megtalált levélcsúcs nem a gyökércsúcs:
- Töröljük a megfelelő kulcsot és a hozzá tartozó mutatót a csúcsból.
- Ha a csúcs még tartalmaz elég kulcsot és mutatót, hogy teljesítse az invariánsokat
  - kész vagyunk.

B+ fa törlés: B eset folytatás

- Ha a levélcsúcsban már túl kevés kulcs van ahhoz, hogy teljesítse az invariánsokat, de a következő, vagy a megelőző testvérének több van, mint amennyi szükséges:
  - Osszuk el a kulcsokat közte és a megfelelő testvére között.
  - Javítsuk ki a testvérek szülőjében a két testvérhez tartozó hasító kulcsot a két testvér közül a 2. minimumára.
- Ha a levélcsúcsban már túl kevés kulcs van ahhoz, hogy teljesítse az invariánst, és a következő, valamint a megelőző testvére is a minimumon van, hogy teljesítse az invariánst:
  - egyesítsük egy vele szomszédos testvérével

B+ fa törlés B eset: testvérek egyesítése

- Testvérek egyesítése
  - Ennek során a két testvér közül a (balról jobbra sorrend szerinti) másodikból a kulcsokat és a hozzájuk tartozó mutatókat sorban átmásoljuk az elsőbe, annak eredeti kulcsai és mutatói után
  - Majd a második testvért töröljük
  - Ezután meg kell ismételnünk a törlő algoritmust a szülőre, hogy eltávolítsuk a szülőből a hasító kulcsot (ami eddig elválasztotta a most egyesített levélcsúcsokat), a most törölt második testvérre hivatkozó mutatóval együtt.

B+ fa törlés: C eset

C. Belső — a gyökértől különböző — csúcsból való törlés:

- Töröljük a belső csúcs éppen most egyesített két gyereke közti hasító kulcsot és az egyesítés során törölt gyerekére hivatkozó mutatót a belső csúcsból!

- Ha a belső csúcsnak van még $\lceil d/2 \rceil$ gyereke, (hogy teljesítse az invariánsokat)
  - kész vagyunk.
- Ha a belső csúcsnak már túl kevés gyereke van ahhoz, hogy teljesítse az invariánsokat, de a következő, vagy a megelőző testvérének több van, mint amennyi szükséges:
  - osszuk el a gyerekeket és a köztük levő hasító kulcsokat egyenlően közte és a megfelelő testvére között, a hasító kulcsok közé a testvérek közti (a közös szülőjükben lévő) hasító kulcsot is beleértve!

B+ fa törlés: C eset

  - A gyerekek és a hasító kulcsok újraelosztása során, a középső hasító kulcs a testvérek közös szülőjében a két testvérhez tartozó régi hasító kulcs helyére kerül úgy, hogy megfelelően reprezentálja a köztük megváltozott vágási pontot!
  - (Ha a két testvérben a gyerekek összlétszáma páratlan, akkor az újraelosztás után is annak a testvérnek legyen több gyereke, akinek előtte is több volt!)
- Ha a belső csúcsnak már túl kevés gyereke van ahhoz, hogy teljesítse az invariánst, és a következő, valamint a megelőző testvére is a minimumon van, hogy teljesítse az invariánst:
  - egyesítsük egy vele szomszédos testvérével!
  - Az egyesített csúcsot a két testvér közül a (balról jobbra sorrend szerinti) elsőből hozzuk létre

B+ fa törlés

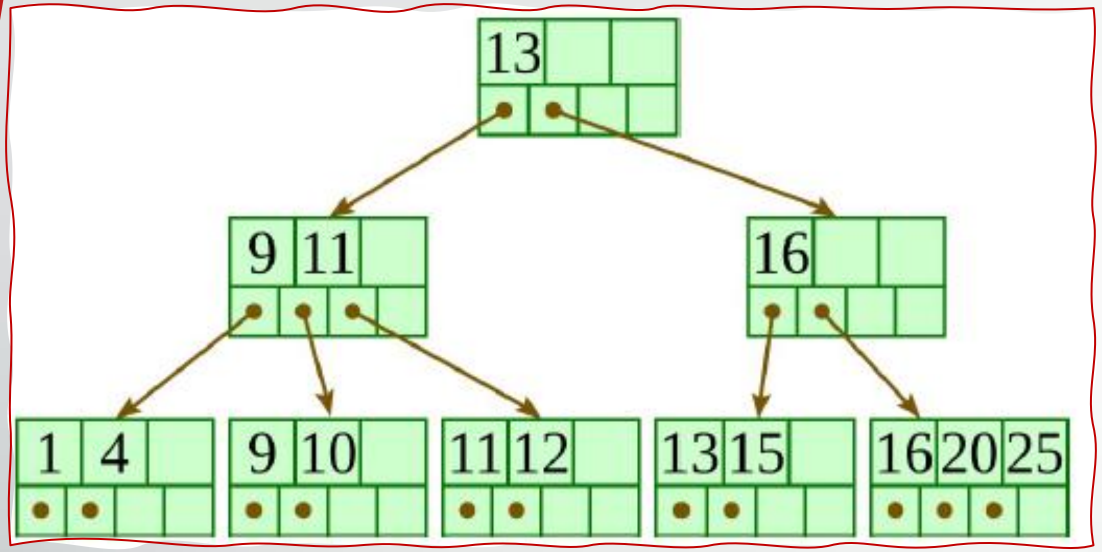
- Gyerekei és hasító kulcsai:
  - először a saját gyerekei és hasító kulcsai az eredeti sorrendben,
  - amiket a két testvér közti (a közös szülőjükben lévő) hasító kulcs követ,
  - és végül a második testvér gyerekei és hasító kulcsai jönnek, szintén az eredeti sorrendben.
- Ezután töröljük a második testvért.
- A két testvér egyesítése után meg kell ismételnünk a törlő algoritmust a közös szülőjükre,
  - hogy eltávolítsuk a szülőből a hasító kulcsot (ami eddig elválasztotta a most egyesített testvéreket), a most törölt második testvérre hivatkozó mutatóval együtt.

B+ fa törlés: D eset

D. A gyökércsúcsból való törlés, ha az nem levélcsúcs

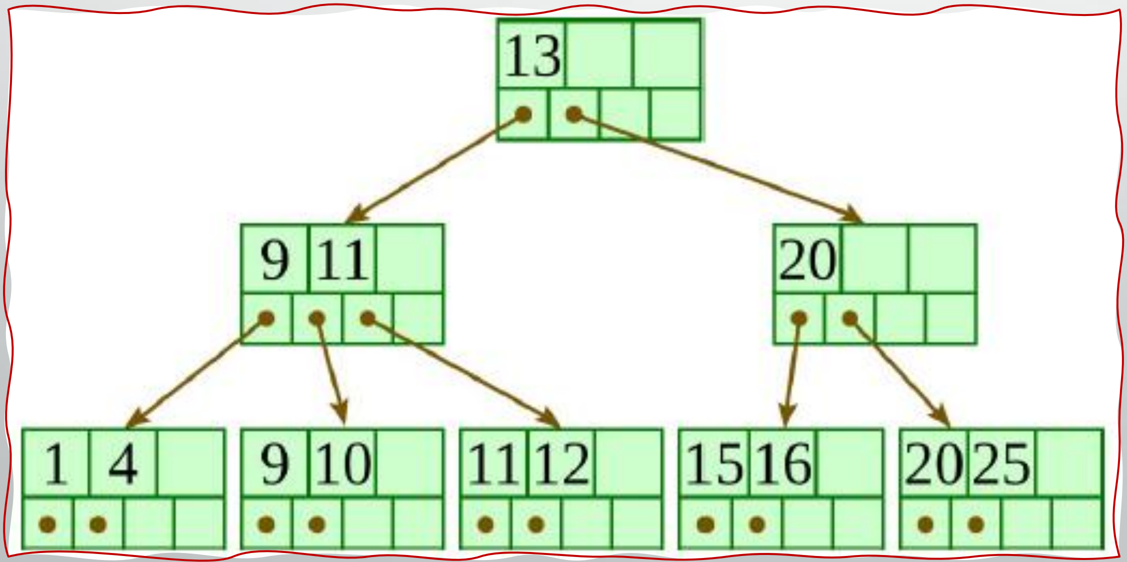
- Töröljük a gyökércsúcs éppen most egyesített két gyereke közti hasító kulcsot és az egyesítés során törölt gyerekére hivatkozó mutatót a gyökércsúcsból!
- Ha a gyökércsúcsnak van még 2 gyereke
  - kész vagyunk.
- Ha a gyökércsúcsnak csak 1 gyereke maradt
  - töröljük a gyökércsúcsot
  - és a megmaradt egyetlen gyereke legyen az új gyökércsúcs!
  - (Ekkor a B+ fa magassága csökken.)

B+ fa törlés példa



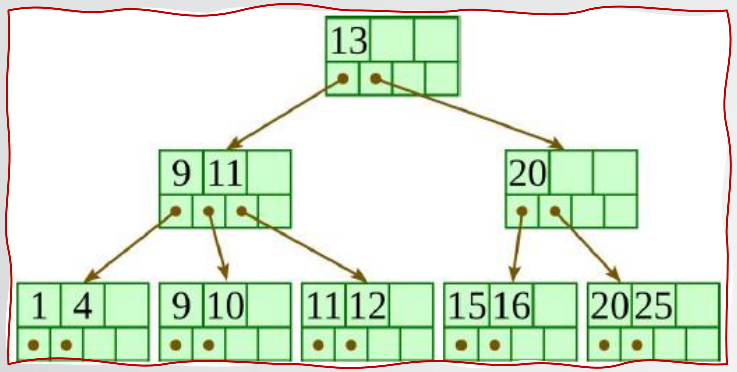
Del (13)

A 15 egyedül marad,
A 16-ot átveszi a jobboldali testvérétől,
Majd a szülőjükben átíródik a megfelelő hasító kulcs.



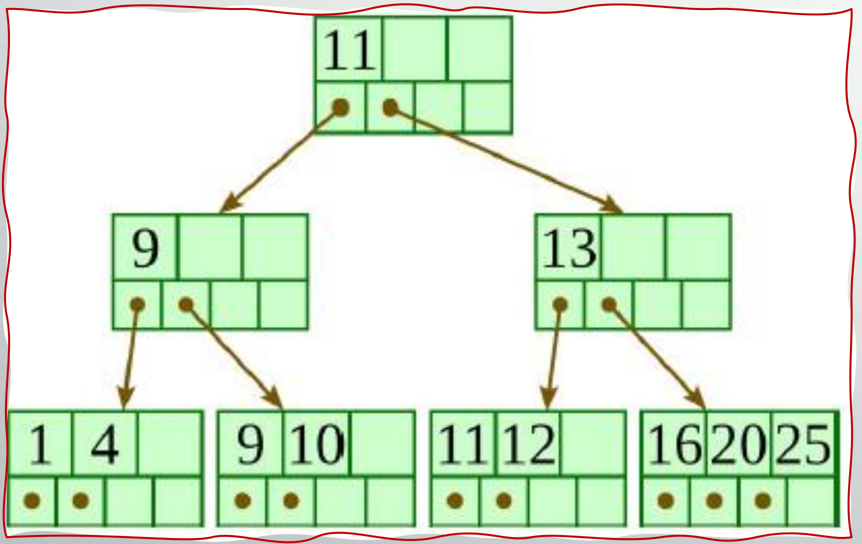
Del (15)

B+ fa törlés példa



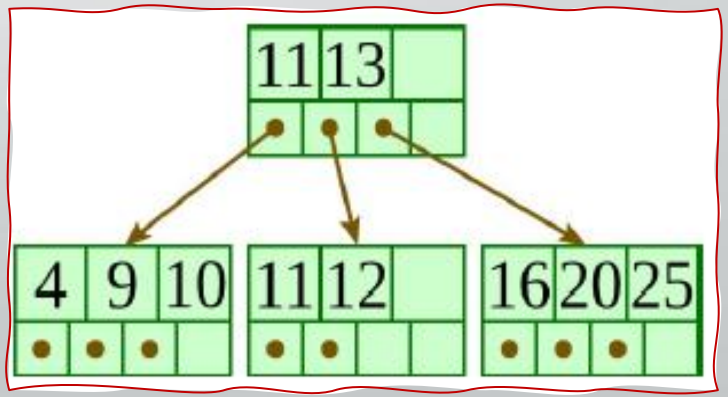
Del (15)

A két testvér levél egyesül,
A 20 hasító kulcs törlődik a szülőből, aminek 1 gyereke marad,
és egyet átvesz a testvérétől;
A 13 a nagyszülőből lejön, és a 11 a 13 helyére megy.



A (4; 9; 10) egyesül,
A szülők egyesülnek,
11 lejön az egyesül szülőbe,
A gyökérnek 1 gyereke marad: törlődik.

Del (1)



Ellenőrző kérdések

1. A *d*-edfokú B+ fák **belső** csúcsainak milyen tulajdonságát ismeri?
2. A *d*-edfokú B+ fák **levél** csúcsainak milyen tulajdonságát ismeri?
3. Adott a { [(1 2) 3 (5 6) 8 (9 10 11) 12 (12 13)] 14 [(15 16 17) 18 (19 20)] } negyedfokú B+fa.
   - Rajzolja le a fát!
   - Szemléltessünk a 8 beszúrását, valamint a 2 és a 14 törlését, **mindhárom esetben az eredeti fára**!
4. Tegyük fel, hogy egy *d*-edfokú B+ fában egy *n* méretű kulcshalmazt tároltunk! Adjon alsó és felső becslést a fa *h* magasságára!

Köszönöm a figyelmet!

Pusztai Kinga

A bemutató Ásványi Tibor: Algoritmusok és adatszerkezetek II. Előadásjegyzete (B+ Fák)
és Fekete István: Algoritmusok és adatszerkezetek / B-Fák
előadásjegyzete alapján készült.