B+ fák

Az alábbiakban Dr. Carl Burch B+-trees című Internetes tananyagának kissé bővített fordítását adjuk.

Tartalom:

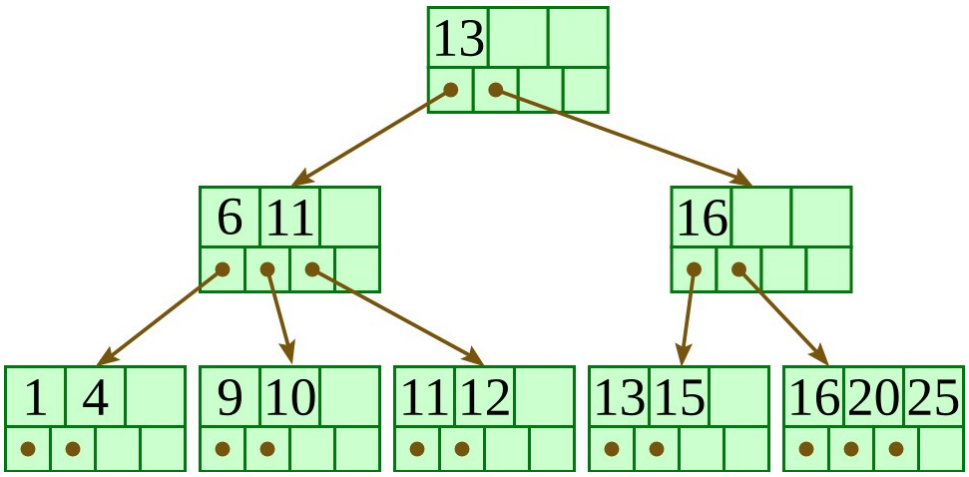
1. Mi a B+ fa?

Az adathalmazokon a legtöbb keresés és módosítás gyorsabban, hatékonyabban hajtható végre, ha a típusértékeket rendezve tároljuk. Nem praktikus azonban a rekordokat szekvenciálisan elhelyezni, mert így a legtöbb beszúrás és törlés kapcsán a tároló jelentős részének újraírása válna szükségessé. A rendezett vagy rendezetlen láncolt listákon pedig a keresés nem elég hatékony.

Ez arra vezet bennünket, hogy az adatokat keresőfába rendezve képzeljük el. Az első ötletünk az AVL fák vagy a piros-fekete fák alkalmazása lehetne, most azonban az adatokat egy véletlen elérésű háttértáron, pl. egy mágneslemezen kívánjuk elhelyezni. A mágneslemezek pedig úgy működnek, hogy egyszerre az adatok egy egész blokkját, tipikusan 512 byte vagy négy kilobyte mennyiségű adatot mozgatunk. Egy bináris keresőfa egy csúcsa ennek csak egy töredékét használná, ezért olyan struktúrát keresünk, ami jobban kihasználja a mágneslemez blokkjait.

Innét jön a B+ fa, amiben minden csúcs legfeljebb d mutatót ($4 \le d$), és legfeljebb $d-1$ kulcsot tartalmaz, ahol d a fára jellemző állandó, *a B+ fa fokszáma*. Úgy tekintjük, hogy a belső csúcsokban mindegyik referencia két kulcs "között" van, azaz 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).

Íme $d=4$ fokszámra egy elég kicsi fa:



Az adatok a levélszinten vannak. A belső kulcsok csak *hasító kulcsok*. Egy adott kulcsú adat keresése során ezek alapján tudhatjuk, melyik ágon keressünk tovább. A levélszinten minden kulcshoz tartozik egy mutató, ami a megfelelő adatrekordra hivatkozik. (A leveleket a d-edik mutatókkal gyakran listába fűzik.)

A B+ fák esetében megköveteljük, hogy a gyökércsúcstól mindegyik levél azonos távolságra legyen. Így a fenti ábrán látható fában tárolt 11 kulcs bármelyikére rákeresve (amelyek mindegyikét a legalsó, a levél szinten találjuk: a belső csúcsok kulcsai csak a tájékozódást szolgálják), három csúcsot érintünk (a gyökércsúcsot, egyik középső szintű csúcsot, és az egyik levelet).

A gyakorlatban persze d sokkal nagyobb. Tegyük fel például, hogy egy-egy blokk 4KB, a kulcsaink 4 byte-os egész számok, és mindegyik mutató egy 6 byte-os relatív cím (a fájl kezdőcíméhez képest). Akkor a d értékét úgy választjuk, hogy a lehető legnagyobb egész szám legyen, amire $4\ (d-1) + 6\ d \leq 4096$. Ezt az egyenlőtlenséget d-re megoldva $d \leq 410$ adódik, tehát a $d=410$ értéket választjuk. Mint láthatjuk, d egészen nagy lehet.

Tetszőleges d-ed fokú B+ fa a következő invariánsokat teljesíti, ahol $4 \leq d$ 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 ($2 \leq i \leq k$) lévő tetszőleges r kulcsra $Cs.kulcs[i-1] \leq r < Cs.kulcs[i]$.
- 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 $floor(d / 2)$ gyereke van. ($floor(d / 2) = d/2$ alsó egész-rész.)
- Minden levél legalább $floor(d / 2)$ 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.

A belső csúcsokban található *hasító kulcsok* segítségével tetszőleges levélcsúcsbeli kulcsot gyorsan megtalálhatunk (illetve megtudhatjuk, ha nincs a levelekben), a fenti invariánsok alapján: a csúcsokban is logaritmikusan keresve, és mindig a megfelelő ágon továbbhaladva, $O(lg\ n)$ lépésben (ahol n a B+ fával ábrázolt adathalmaz mérete). A hasító kulcsok nem okvetlenül szerepelnek a levelekben. (Mint látni fogjuk, a törlő algoritmus ténylegesen is létrehoz ilyen fákat.)

Ugyan a fa magassága $O(lg\ n)$, a gyakorlatban a fa h magassága az $lg\ n$ érték töredéke:
HF: Bizonyítsuk be, hogy a $log[d](n/(d-1)) \leq h \leq log[floor(d/2)](n/2)$ egyenlőtlenség teljesül!
($log[d](n)$ = az n szám d alapú logaritmusa.)

Ez azt jelenti, hogy a fenti $d=410$ értékkel $log[410](n/409) \leq h \leq log[205](n/2)$.
Pl. ha $n=1.000.000.000$ (n=egymilliárd), akkor $2{,}4 < h < 3{,}8$, vagyis $h=3$, azaz a fának pontosan négy szintje van. Egy ilyen fánál a felső két szintet az adatbázis megnyitásakor betöltjük a központi tárba. A levélszintről viszont még egyet lépünk a tényleges adatrekord eléréséhez. Ez azt jelenti, hogy egy-egy adat eléréséhez összesen háromszor olvasunk a lemezről, *egymilliárd* rekordot tartalmazó adatbázis esetén. Ha pedig a keresett kulcsú rekord nincs az adatbázisban, csak kétszer olvasunk a lemezről.

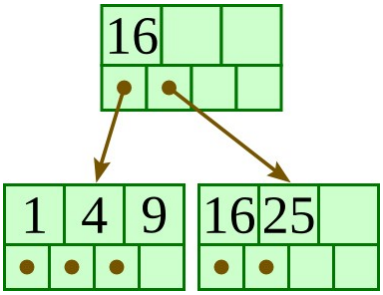
Az alábbi példákban továbbra is a $d=4$ értékkel dolgozunk. A fenti invariánsok alapján ez azt jelenti, hogy minden levél legalább két kulcsot tartalmaz; minden belső csúcsnak pedig legalább két gyereke és legalább egy hasító kulcsa van.

2. A beszúrás algoritmusa

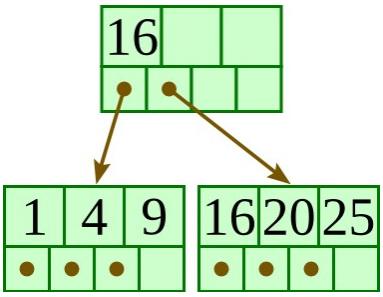
Ha a fa üres, hozzunk létre egy új levélcsúcsot, ami egyben a gyökércsúcs is, és a beszúrandó kulcs/mutató pár a tartalma! Különben keressük meg a kulcsnak megfelelő levelet! Ha a levélben már szerepel a kulcs, a beszúrás sikertelen. Különben menjünk az 1. pontra!

1. Ha a csúcsban van üres hely, szúrjuk be a megfelelő kulcs/mutató párt kulcs szerint rendezetten ebbe a csúcsba!
2. Ha a csúcs már tele van, vágjuk szét két csúccsá, és 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.)

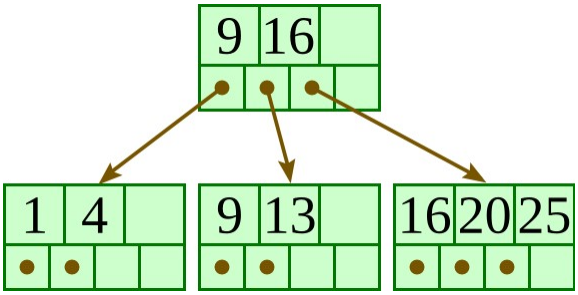
Eredeti:



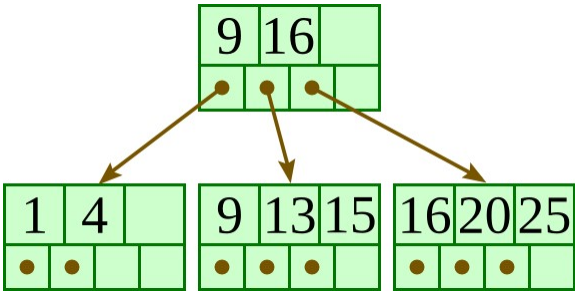
A 20 beszúrása:
Beszúródik a megfelelő levélbe.



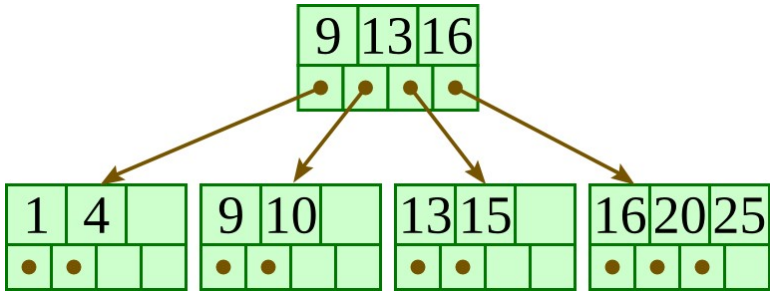
A 13 beszúrása:
(1;4 | 9;13) hasad, a 2. levél minimuma a szülőbe másolódik új hasítókulcsnak.



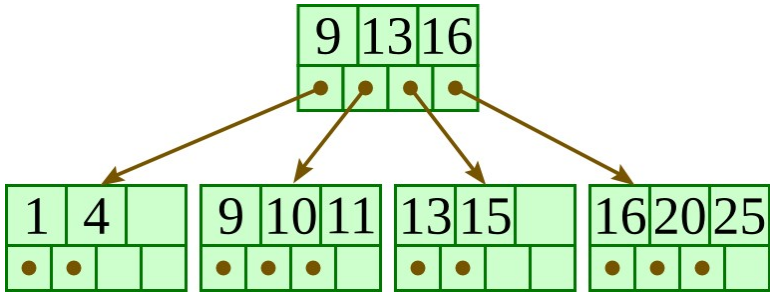
A 15 beszúrása:
Beszúródik a megfelelő levélbe.



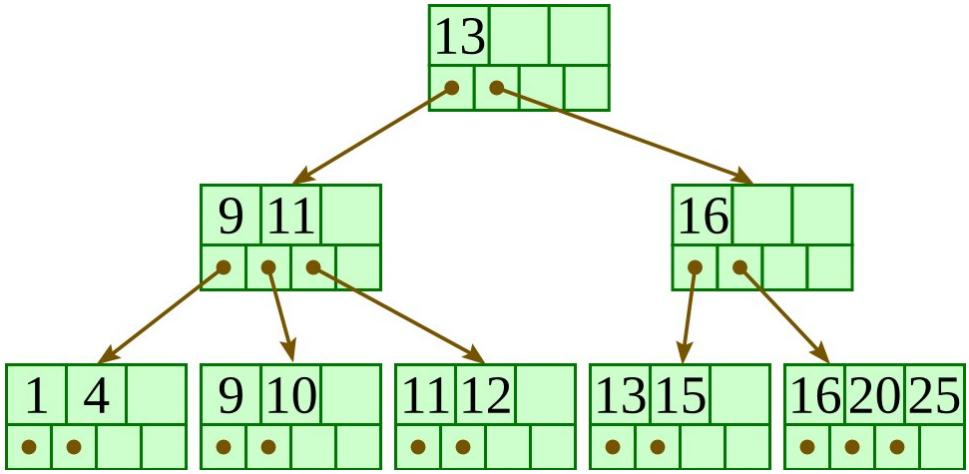
A 10 beszúrása:
(9;10 | 13;15) hasad, (13;15) minimuma a szülőbe másolódik új hasítókulcsnak.



A 11 beszúrása:
Beszúródik a megfelelő levélbe.



A 12 beszúrása:
(9;10 | 11;12) hasad, (11;12) minimuma a szülőbe másolódik új hasítókulcsnak; (9;11 | 13;16) hasad, (13;16) minimuma felmegy az újonnan létrehozott gyökérbe.



3. A törlés algoritmusa

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

A, A keresés során megtalált levélcsúcs egyben a gyökércsúcs is:

1. Töröljük a megfelelő kulcsot és a hozzá tartozó mutatót a csúcsból!
2. Ha a csúcs tartalmaz még kulcsot, kész vagyunk.
3. Különben töröljük a fa egyetlen csúcsát, és üres fát kapunk.

B, A keresés során megtalált levélcsúcs nem a gyökércsúcs:

1. Töröljük a megfelelő kulcsot és a hozzá tartozó mutatót a levélcsúcsból!
2. Ha a levélcsúcs még tartalmaz elég kulcsot és mutatót, hogy teljesítse az invariánsokat, kész vagyunk.
3. 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 egyenlően közte és a megfelelő testvére között! Írjuk át a két testvér közös szülőjében a két testvérhez tartozó hasító kulcsot a két testvér közül a második minimumára!

4. 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, akkor egyesítsük egy vele szomszédos testvérével! 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.

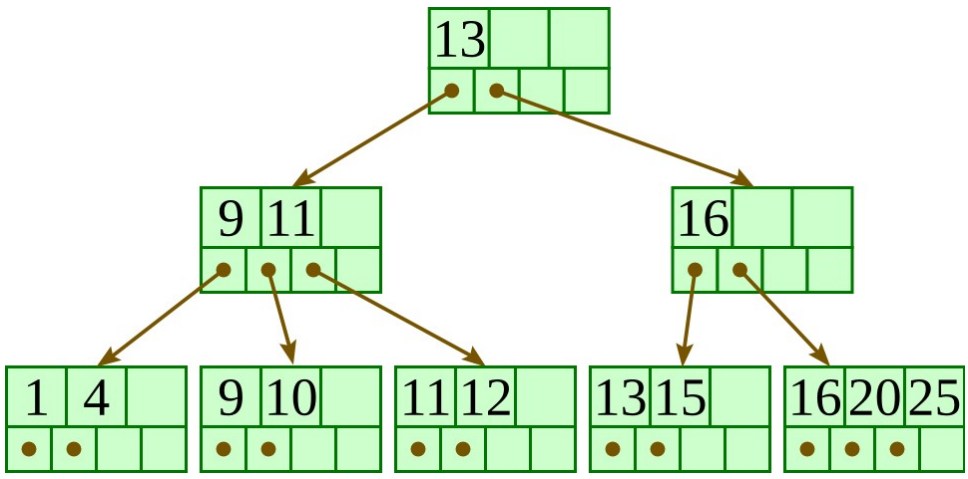
C, Belső — a gyökértől különböző — csúcsból való törlés:

1. Töröljük a belső csúcs éppen most egyesített két gyereke közti hasító kulcsot és az egyesítés során törölt gyerekére hivatkozó mutatót a belső csúcsból!
2. Ha a belső csúcsnak van még floor(d/2) gyereke, (hogy teljesítse az invariánsokat) kész vagyunk.
3. 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! 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!)
4. 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, akkor egyesítsük egy vele szomszédos testvérével! Az egyesített csúcsot a két testvér közül a (balról jobbra sorrend szerinti) elsőből hozzuk létre. 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.

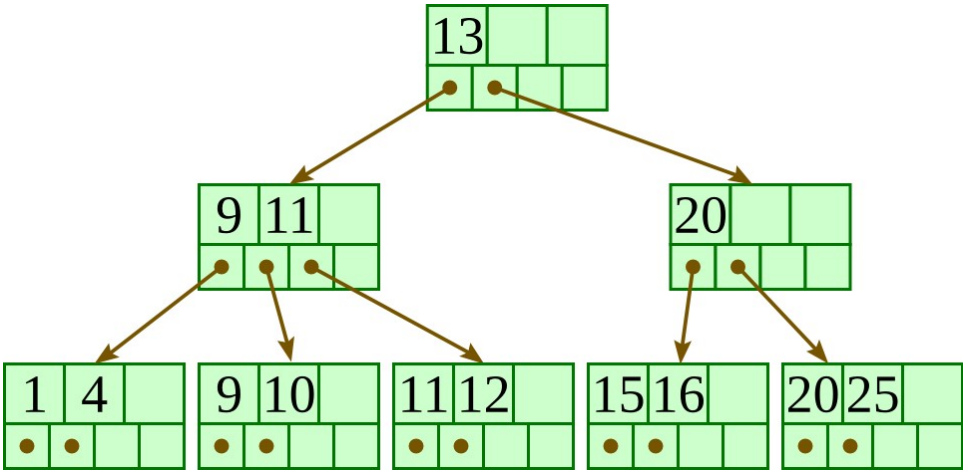
D, A gyökércsúcsból való törlés, ha az nem levélcsúcs:

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

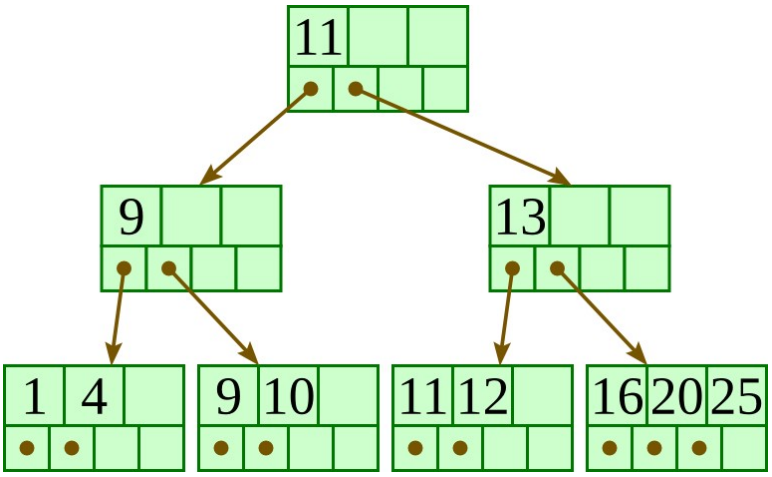
Eredeti:



A 13 törlése:
A 15 egyedül marad, a 16-ot átveszi a jobboldali testvérétől, majd a szülőjükben átíródik a megfelelő hasító kulcs.



A 15 törlése:
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.



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