

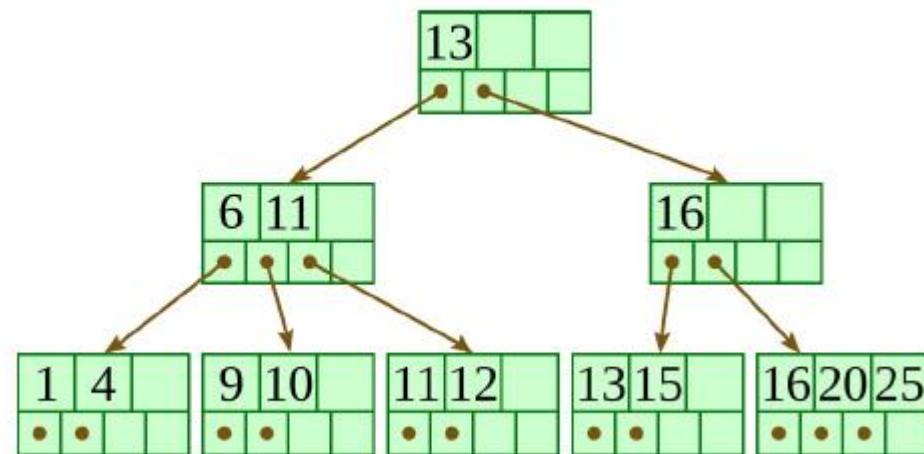
Algoritmusok és adatszerkeztek II.

2. gyakorlat

Mi a B+ fa?

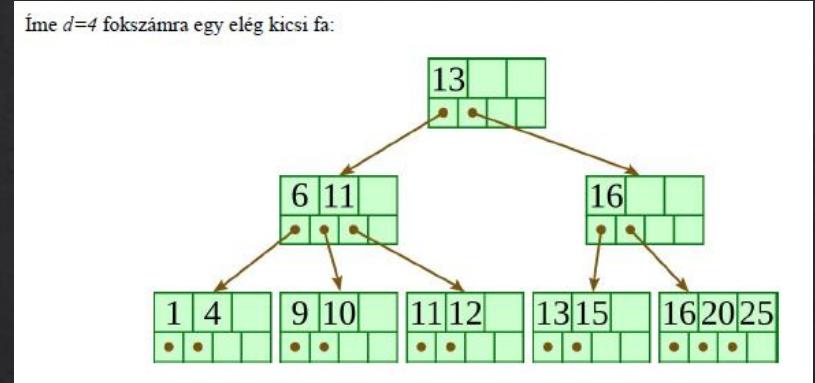
- ◊ Adatbázis kezelők használják, több millió rekord elérését teszik lehetővé pár lépésben egy „nagyon gyorsan terebélyesedő” fával.
- ◊ d=4 esetben 3 kulcs és 4 pointer van minden Node-ban.
- ◊ A belső pontok a keresést (beszúrást, törlést) irányítják.
- ◊ Levelek azonos szinten vannak, a levelek az adott kulcshoz tartozó rekordra mutatnak.
- ◊ Gyakorlatban a d igen nagy érték, például 4 Kbyte blokkméret esetén d=410, ekkor egy 4 szintű fa 1 milliárd rekordot tud megcímézni.
- ◊ A felső két szintet a memóriában tartják, majd 3 lemez olvasó művelettel (2 szint a fában + a konkrét rekord beolvasása) elérhetjük a rekord adatait.

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

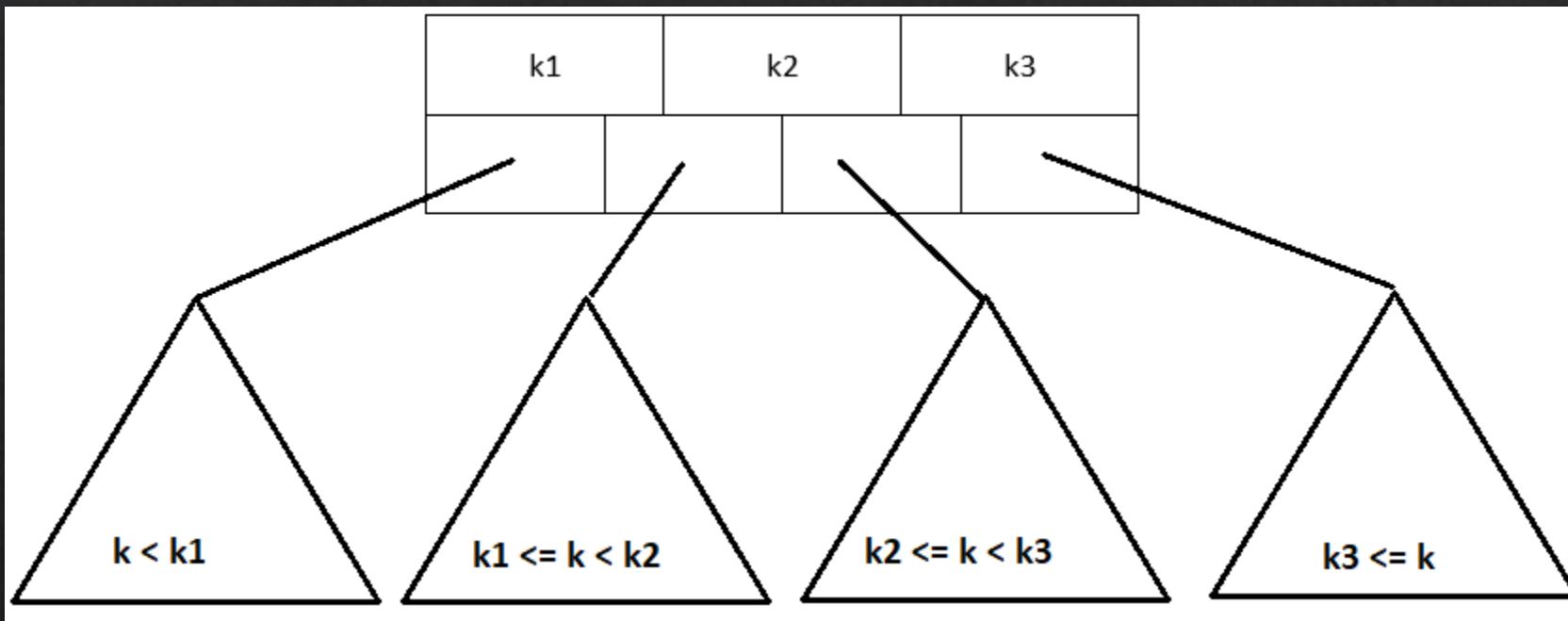


Belső csúcsok tulajdonságai

- ❖ minden csúcs legfeljebb d mutatót ($4 \leq d$), és legfeljebb $d-1$ kulcsot tartalmaz.
(d : állandó, a B+ fa fokszáma)
- ❖ 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
- ❖ így az i -edik gyerekhez tartozó részfában ($2 \leq i \leq k$) lévő tetszőleges r kulcsra:
 $\text{Cs.kulcs}[i-1] \leq r < \text{Cs.kulcs}[i]$.
- ❖ 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 $\lfloor d/2 \rfloor$ gyereke van.

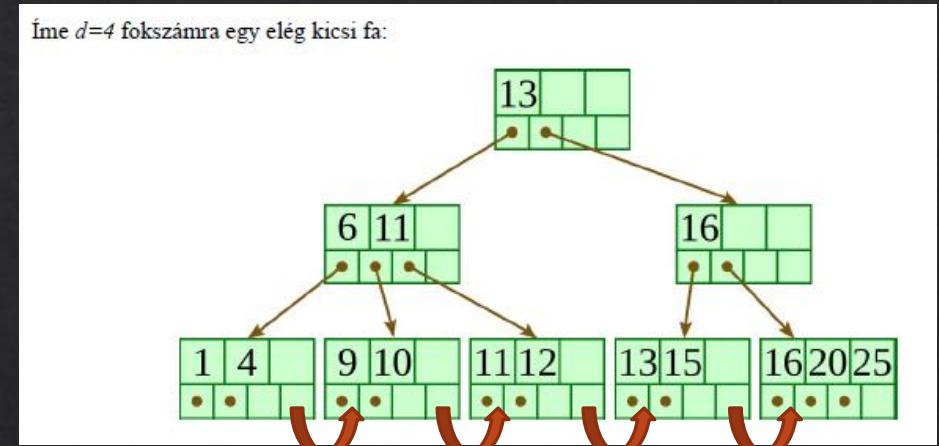


Keresés irányítása belső pontok esetén $d=4$ esetben



Levelek tulajdonságai

- ◆ minden levélben legfeljebb $d-1$ kulcs, és ugyanennyi, a megfelelő (azaz ilyen kulcsú) adatrekordra hivatkozó mutató található.
- ◆ A gyökértől mindegyik levél ugyanolyan távol található.
- ◆ 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.
- ◆ A levelekben található „felesleges mutatót” fel szokták használni arra, hogy a leveleket egy listává fűzik össze, így lehetővé válik a kulcs szerinti rendezett sorrendű feldolgozás.

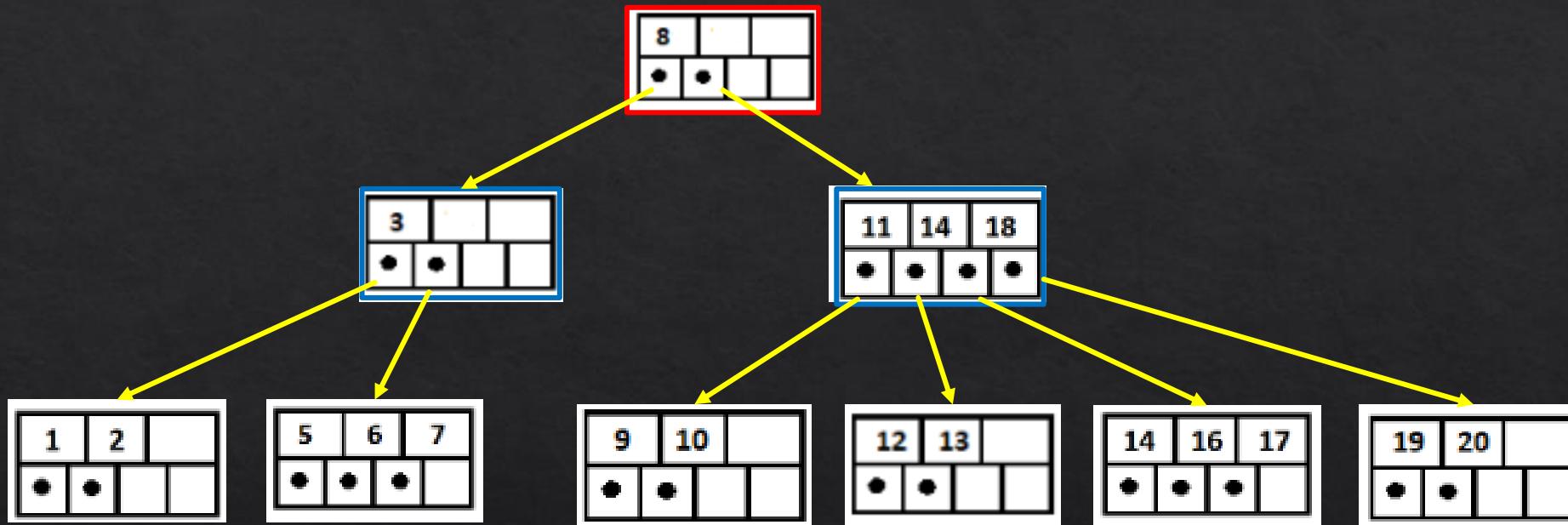


Feladat

- ❖ 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, tehát
 - ❖ $4(d-1) + 6d \leq 4096$
 - ❖ $10d - 4 \leq 4096$
 - ❖ $10d \leq 4100$
 - ❖ $d = 410$ –nek érdemes választani a B+ fa fokszámát

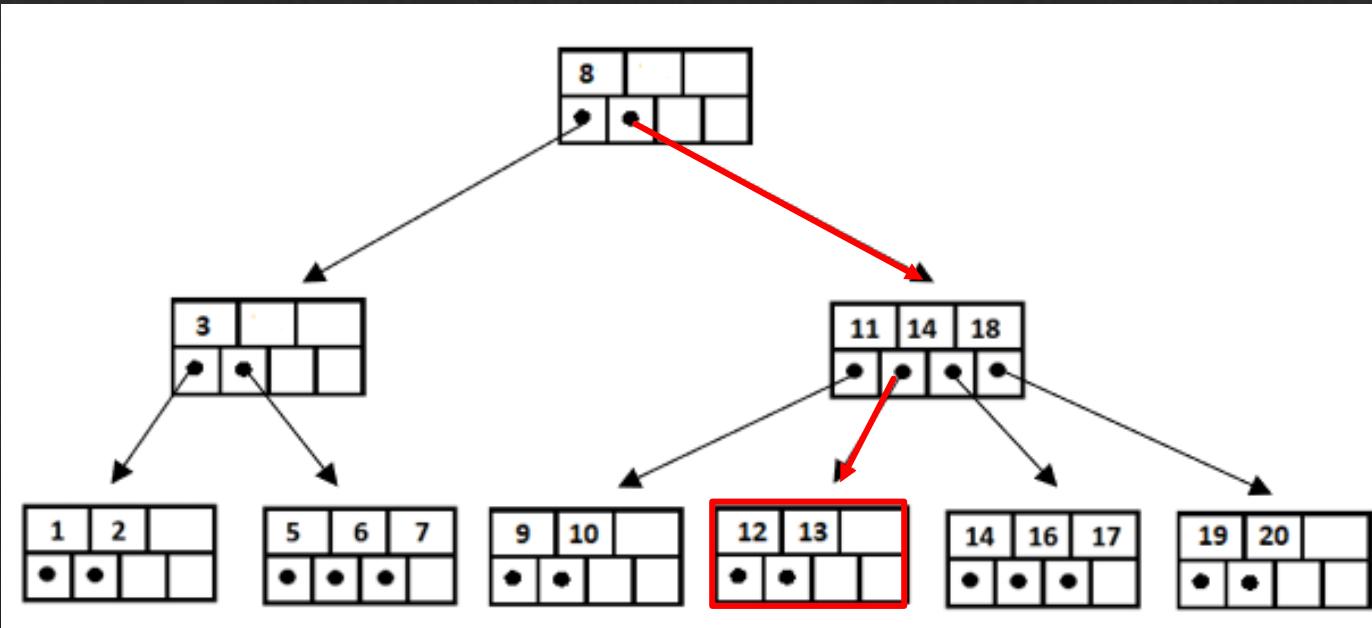
B+ fa zárójeles leírása

- ◊ Rajzolja le a $\{ [(1 \ 2) \ 3 \ (5 \ 6 \ 7)] \ 8 \ [(9 \ 10) \ 11 \ (12 \ 13) \ 14 \ (14 \ 16 \ 17) \ 18 \ (19 \ 20)] \ }$ negyedfokú B+ fát.



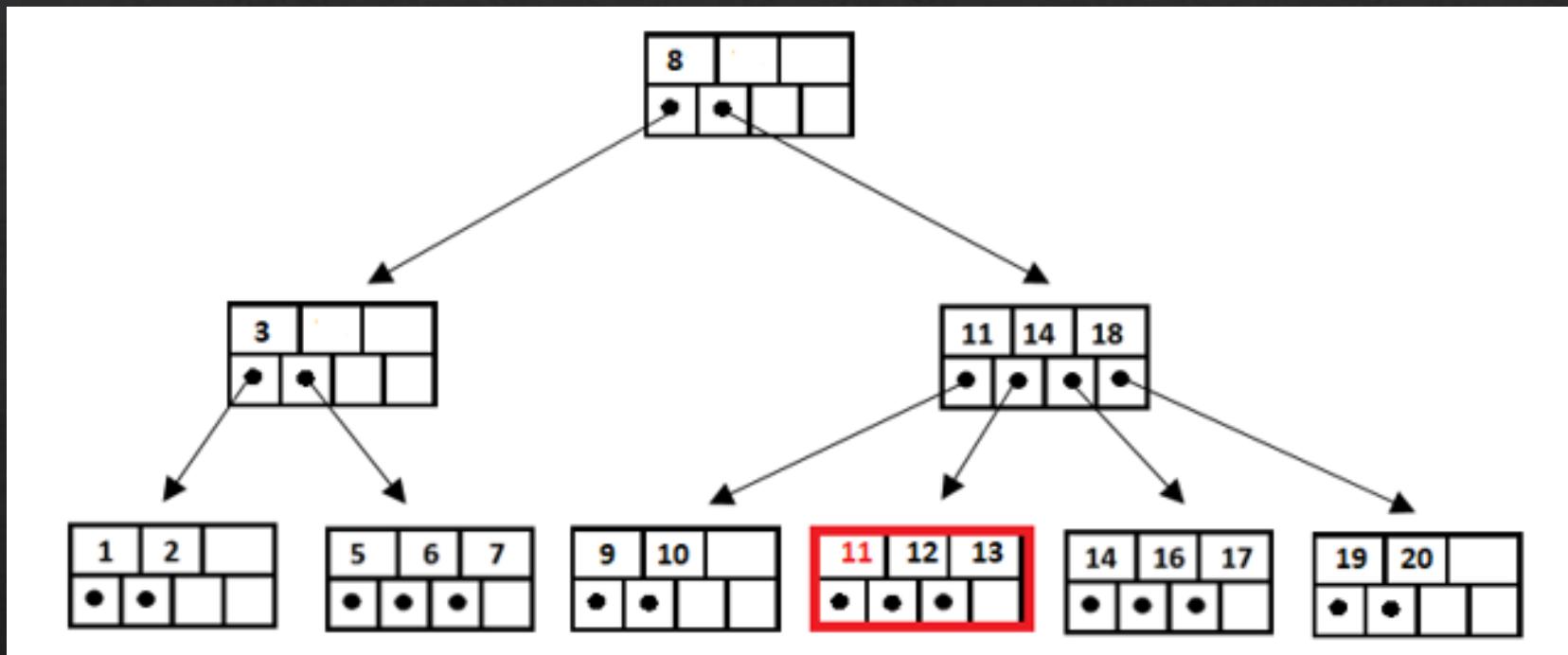
Beszúrás esetei

- ◆ Szűrjuk be a 11 kulcsot:



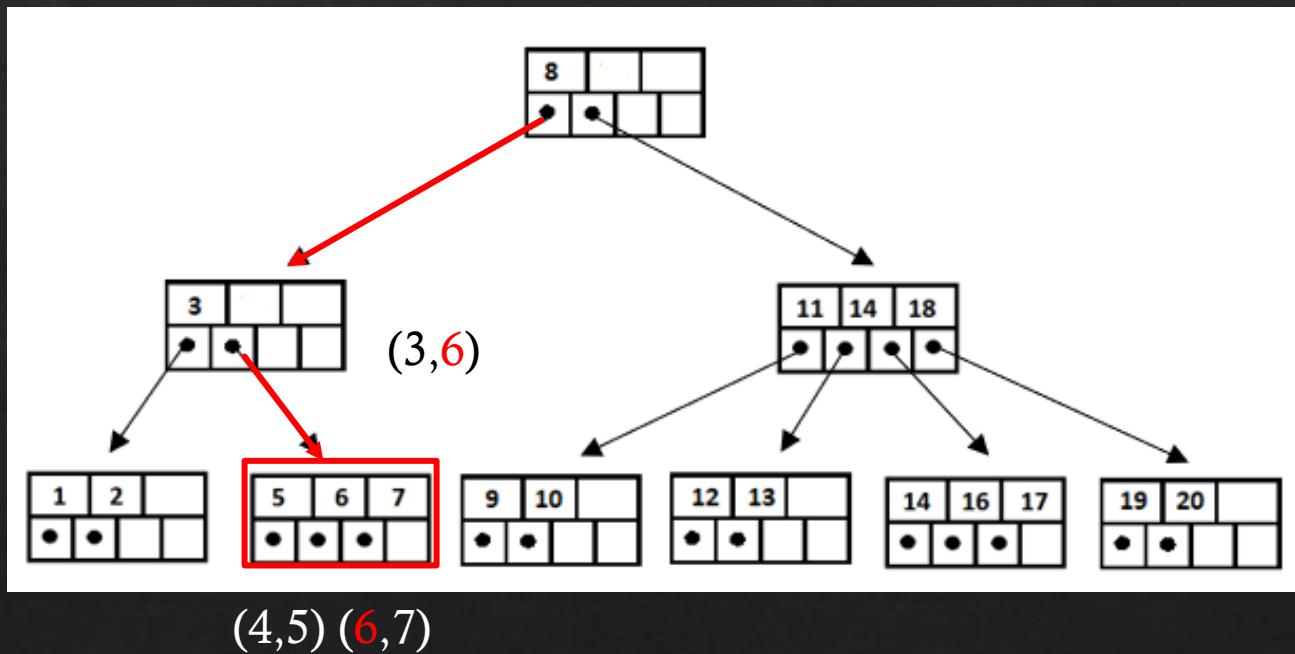
- ◆ Megkeressük a 11-es kulcsot: a (12,13) levélbe illik.
- ◆ Nem szerepel a kulcsok között, és van még egy üres hely, így egyszerűen beszúrjuk.

11 beszúrása után a kapott fa



Beszúrás esetei

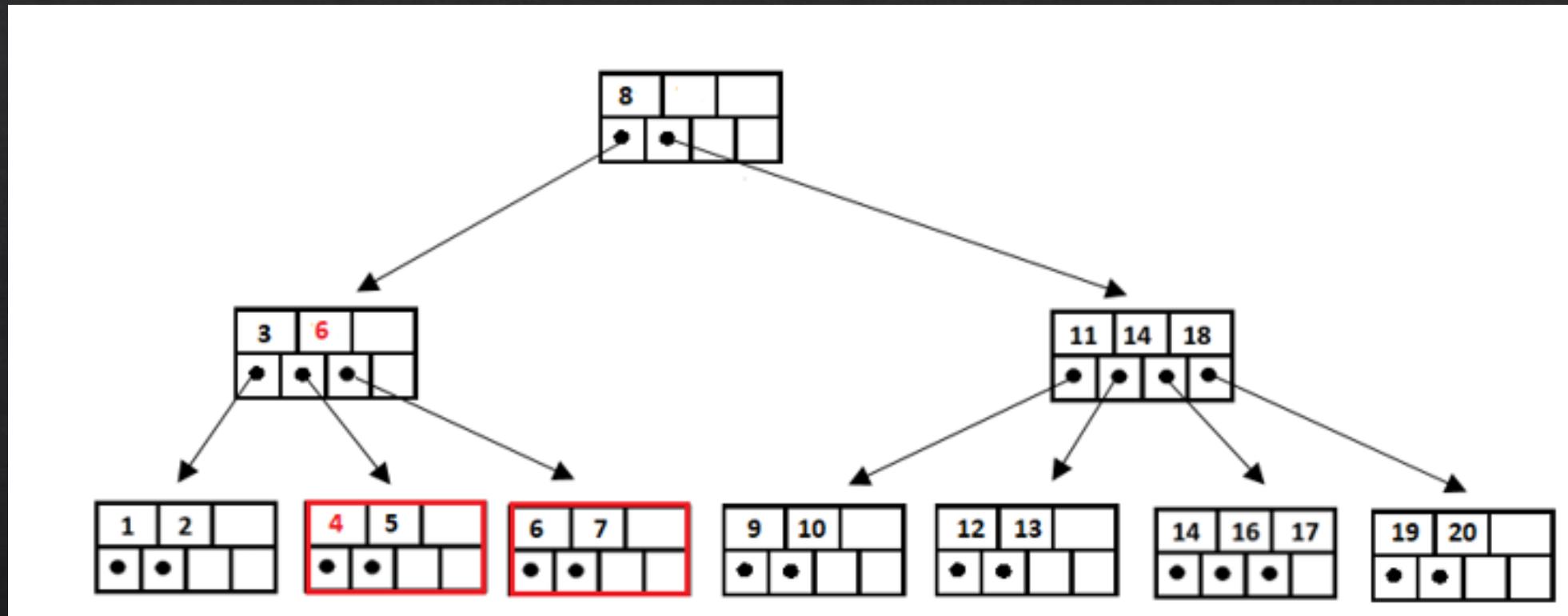
- ◆ Szúrjuk be a 4 kulcsot:



- ◆ Megkeressük a 4-es kulcsot: az (5,6,7) levélbe illik.

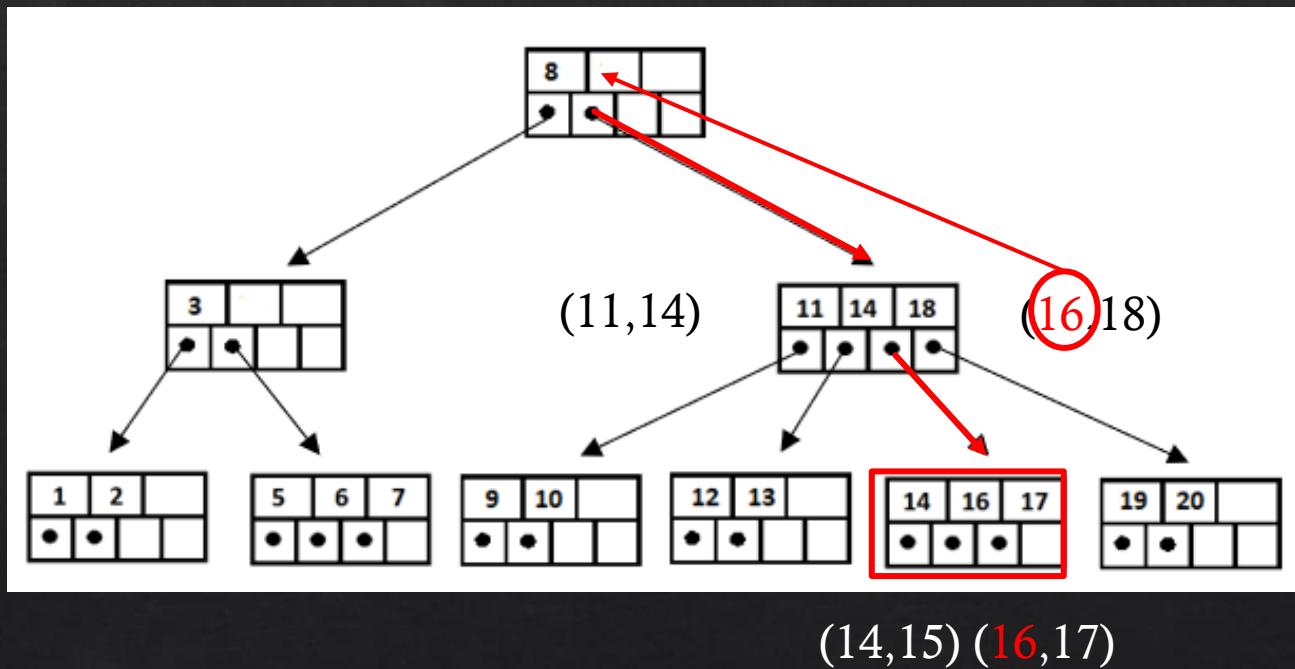
- ◆ Ketté vágjuk a levelet:
- ◆ (4,5) és (6,7) levélre
- ◆ A szülőbe is kell egy új kulcs, de ott van még hely.
- ◆ A kettévágásnál kapott jobb levél legkisebb kulcsát szúrjuk be a szülőbe:
- ◆ (3,6) lesz a szülőben a két kulcs.

4 beszúrása után a kapott fa



Beszúrás esetei

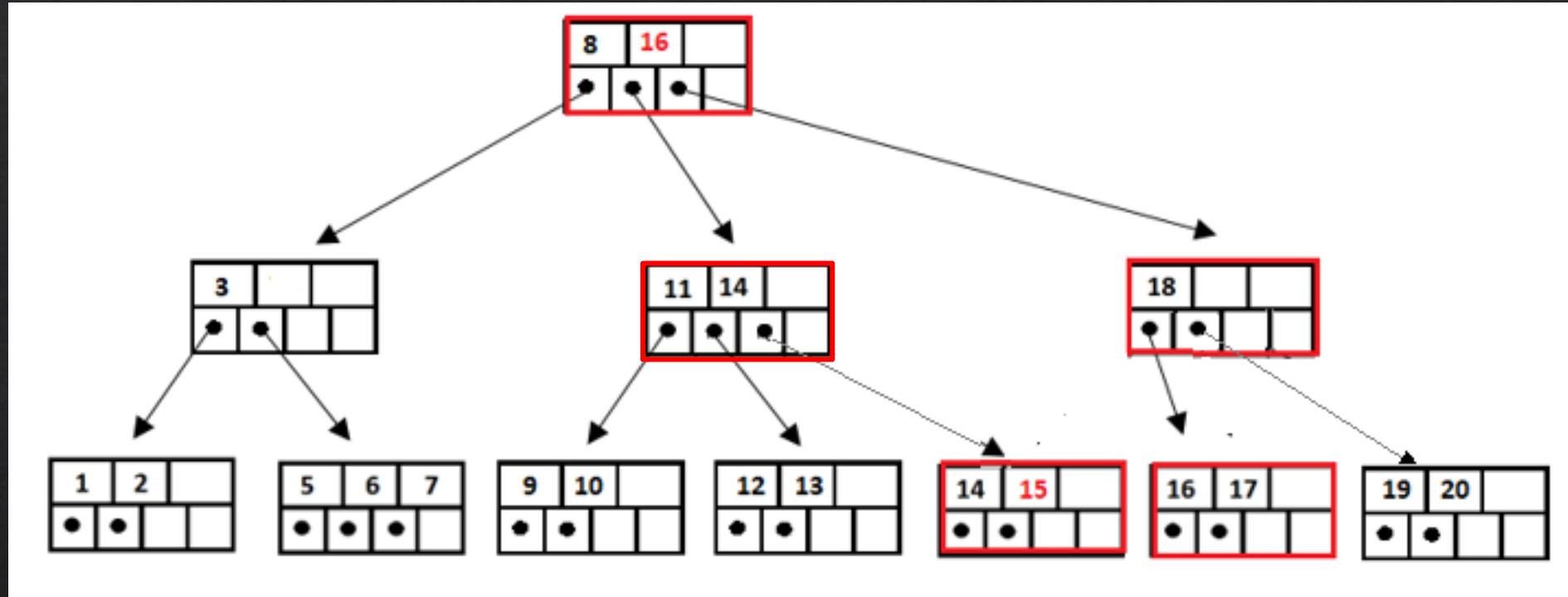
- ◆ Szúrjuk be a 15 kulcsot:



- ◆ Megkeressük a 15-ös kulcsot: az (14,16,17) levélbe illik.

- ◆ Ketté vágjuk a levelet:
 - ◆ (14,15) és (16,17) levélre
 - ◆ A szülőbe is kell egy új kulcs, a 16-ot kellene beszúrni, de már nincs hely.
 - ◆ A szülő csúcs is ketté vágódik: (11,14)(16,18)
 - ◆ A középső kulcsot (16) nem a kettévágott csúcsba, hanem annak szülőjébe szúrjuk be.
 - ◆ A 16 kulcs a fa gyökerébe kerül hasító kulcsként.

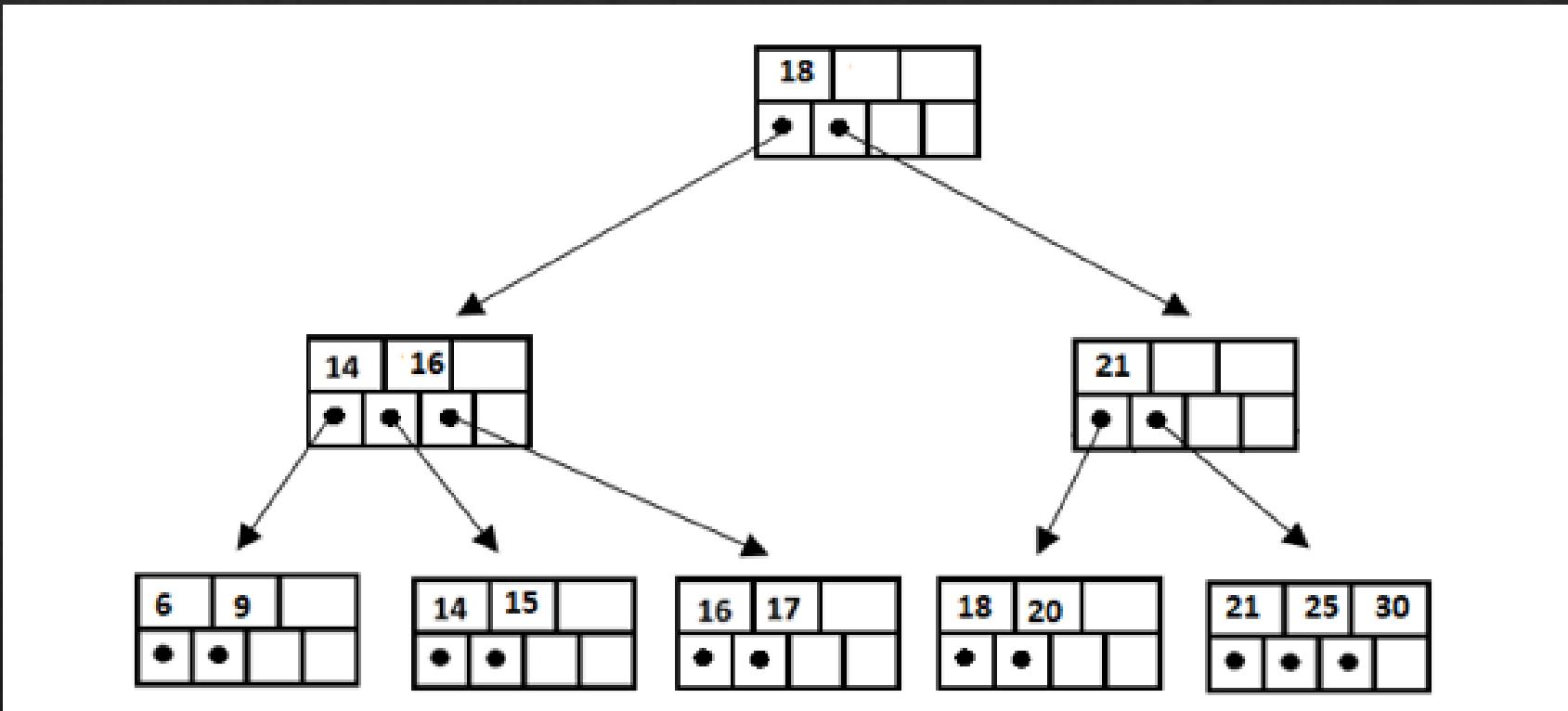
15 beszúrása után a kapott fa



Megjegyzés: ha a fa gyökerébe sem fér már el az új kulcs, akkor a gyökér is a belső pontok szabálya szerint ketté vágódik, és a középső kulcs egy új Node-ba kerül, ez lesz a fa új gyökere, a B+ fa szintjeinek száma eggyel megnő.

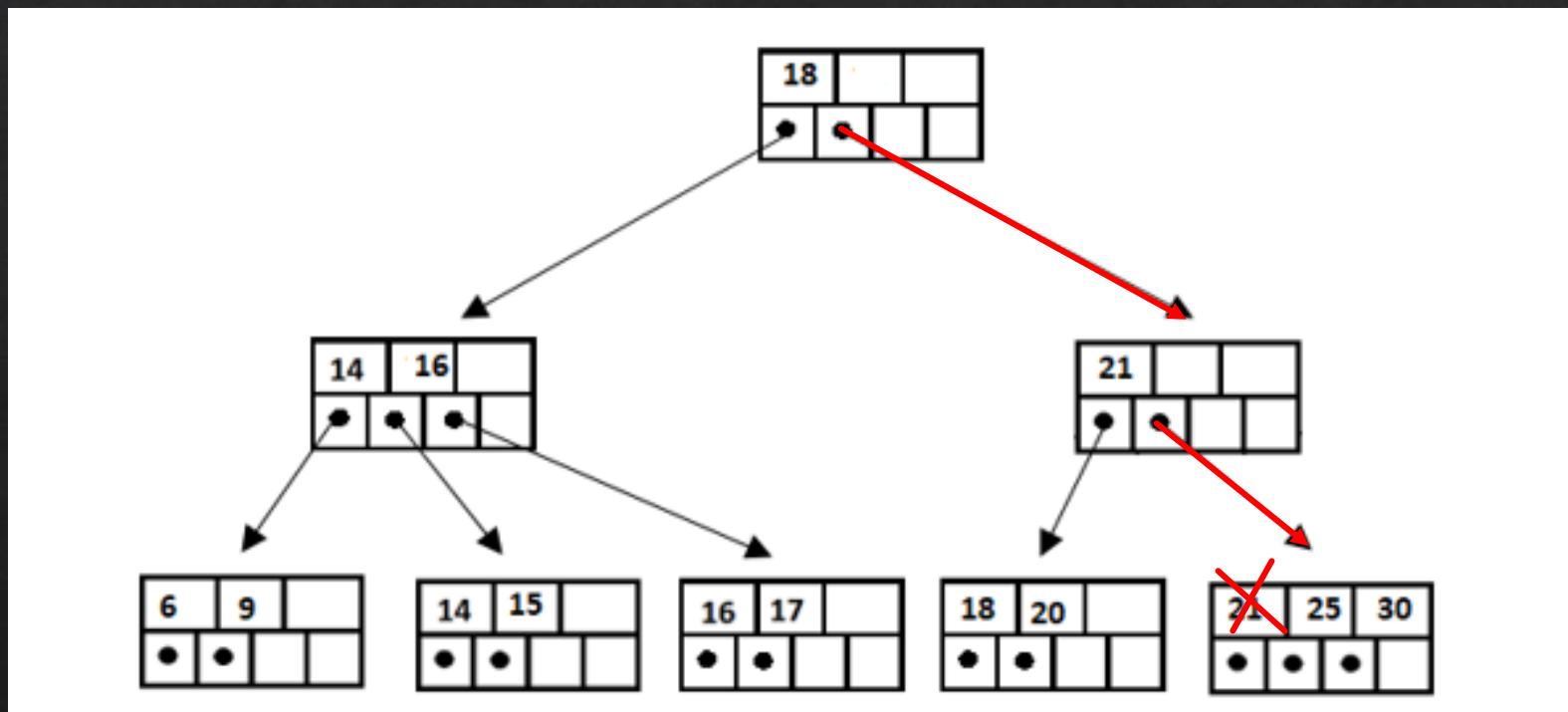
Törlés esetei

- ❖ A törlés eseteit az következő B+ fán fogjuk megvizsgálni:
{ [(6 9) 14 (14 15) 16 (16 17)] 18 [(18 20) 21 (21 25 30)] }
- ❖ Rajzoljuk le a fát!



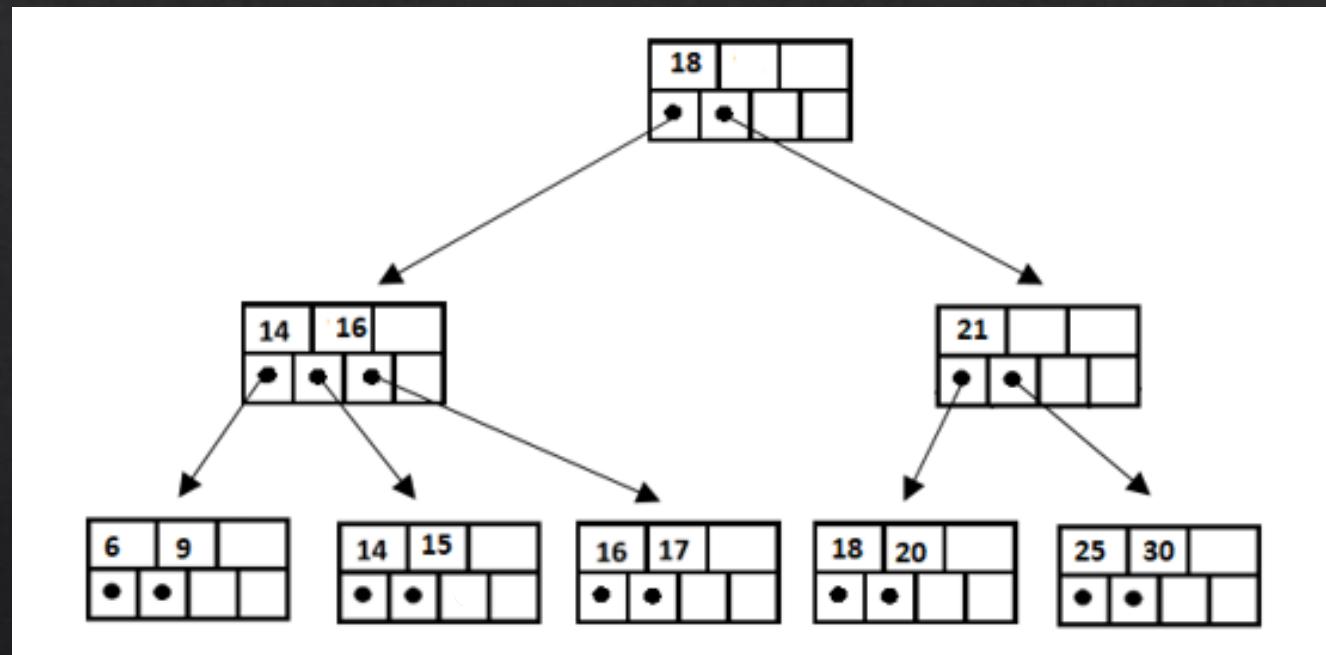
Törlés esetei

- ◆ Legegyszerűbb az az eset, amikor egy 3 kulccsal rendelkező levélből az egyik kulcsot töröljük, például a 21-et.
- ◆ Ilyenkor a levélből törlődik a 21-es kulcs (a hozzá tartozó rekorddal együtt), de a szülőben marad a 21 hasító kulcs.



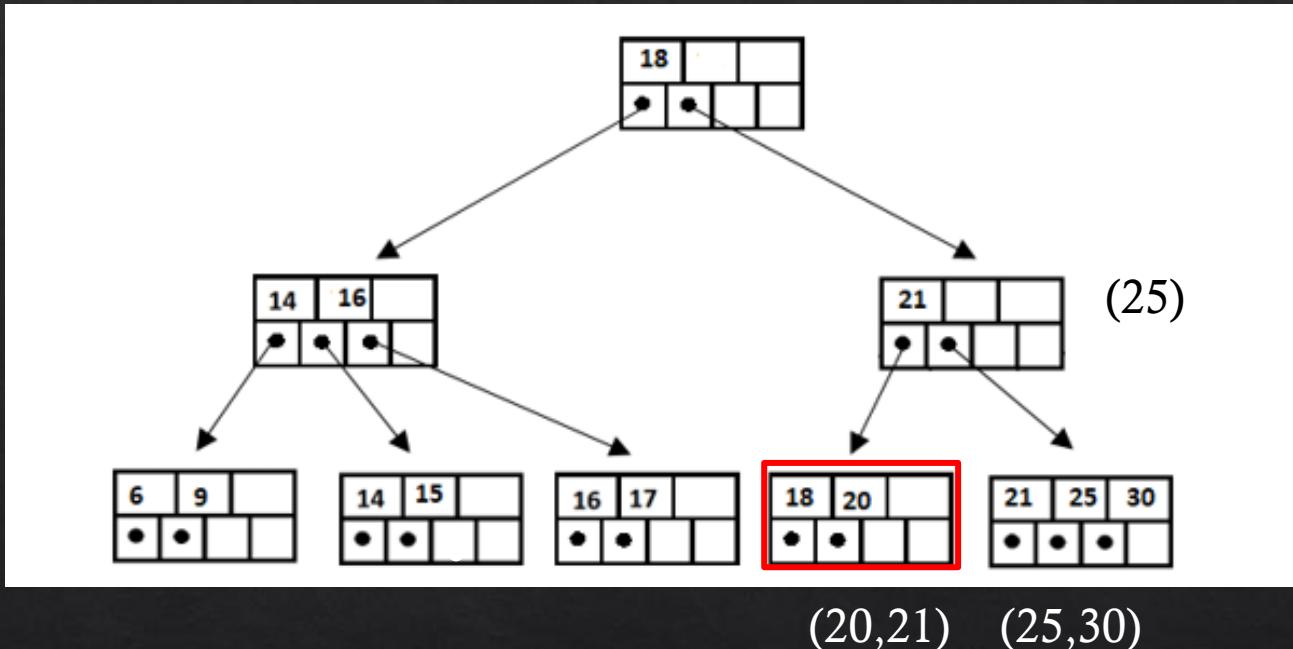
21 törlése után a kapott fa

- ❖ Ilyenkor a levélből törlődik a 21-es kulcs (a hozzá tartozó rekorddal együtt), de a szülőben marad a 21 hasító kulcs.



18 törlése

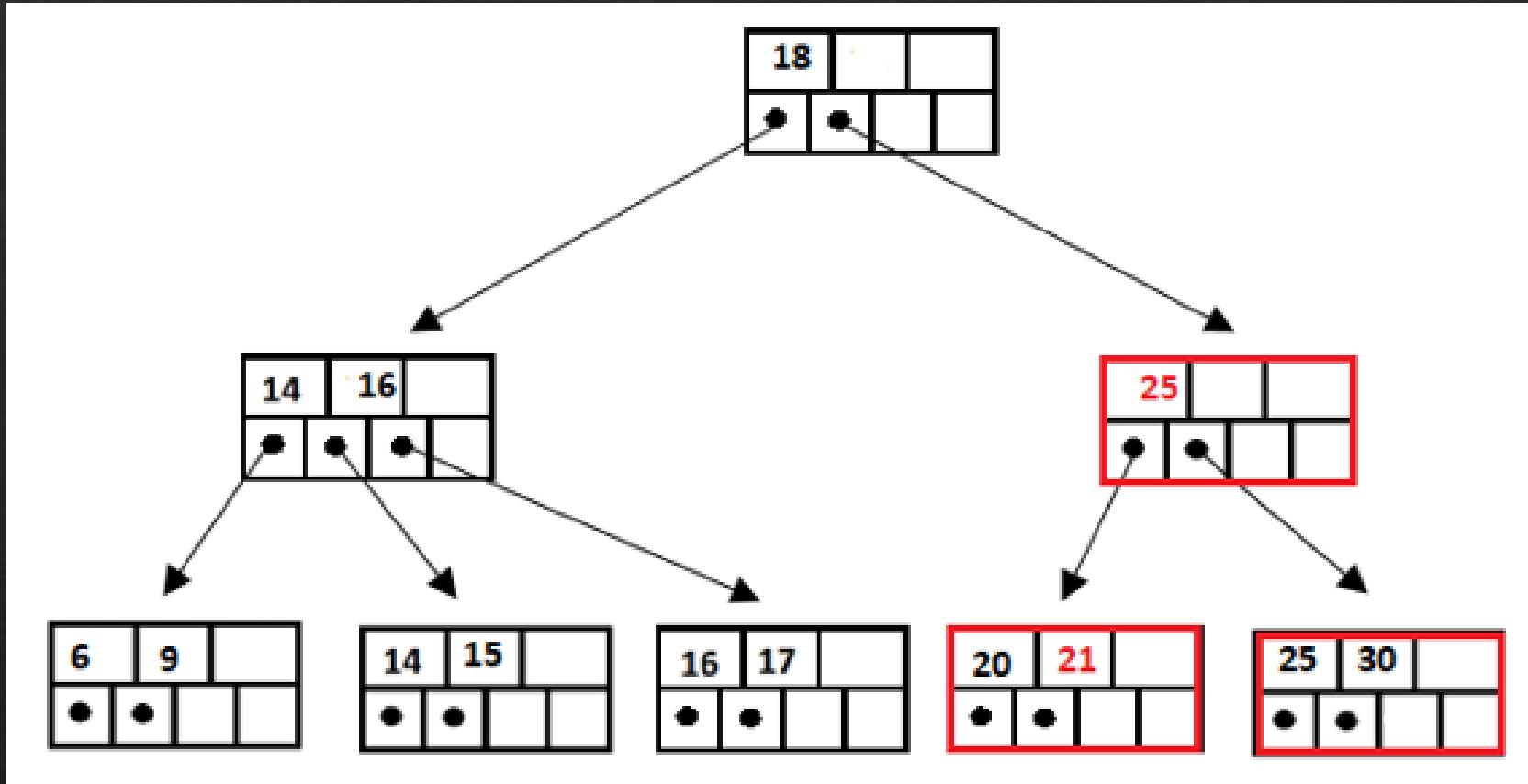
- ❖ Megkeressük a 18-as kulcsot.



- ❖ Ha kitöröljük, a levélben csak a 20-as kulcs marad, ami szabálytalan.

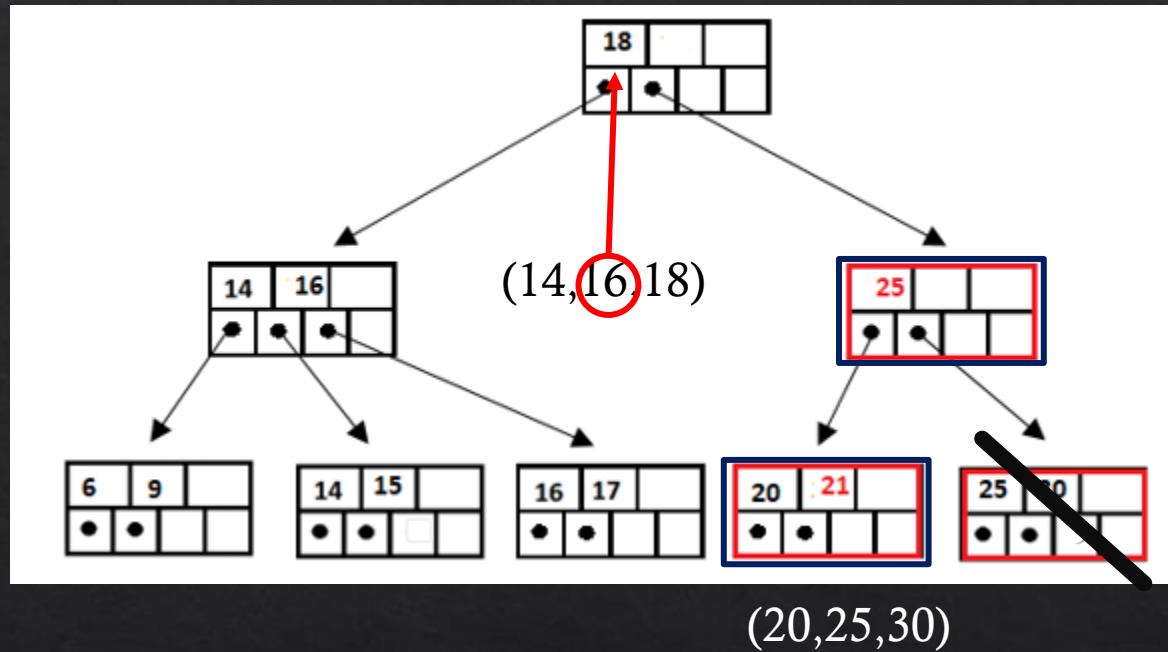
- ❖ Ilyenkor elsőként a bal vagy jobb testvérét vizsgáljuk meg, hátha valamelyik át tud adni egy kulcsot.
- ❖ Jobb testvértől megkapja a 21-es kulcsot.
- ❖ A szülőben 21-es kulcsot ki kell cserélni az új osztási pontnak megfelelően: 25-re.
- ❖ A két testvér közül a második mimumumára kell átírni az eredeti kulcsot a szülőben.

18 törlése után kapott fa



A kapott fából a 21 törlése

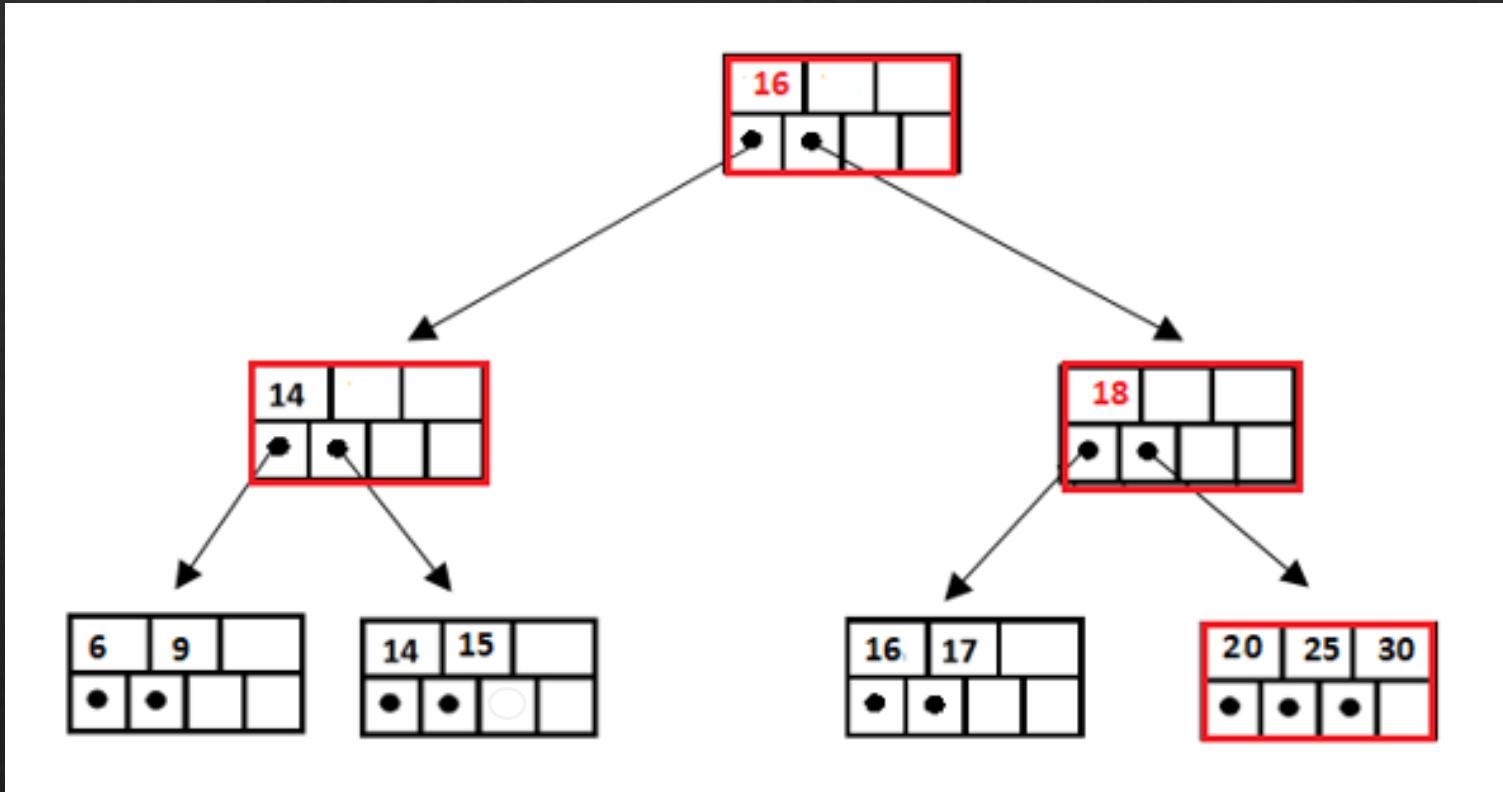
- ❖ Megkeressük a 21-es kulcsot.



- ❖ Ha kitöröljük, a levélben csak a 20-as kulcs marad, ami szabálytalan. De a testvére most nem tudja kisegíteni a csúcsot, így a két Node összevonódik: a jobb oldali Nodeból a kulcsok átmásolódnak, majd a Node megszűnik.

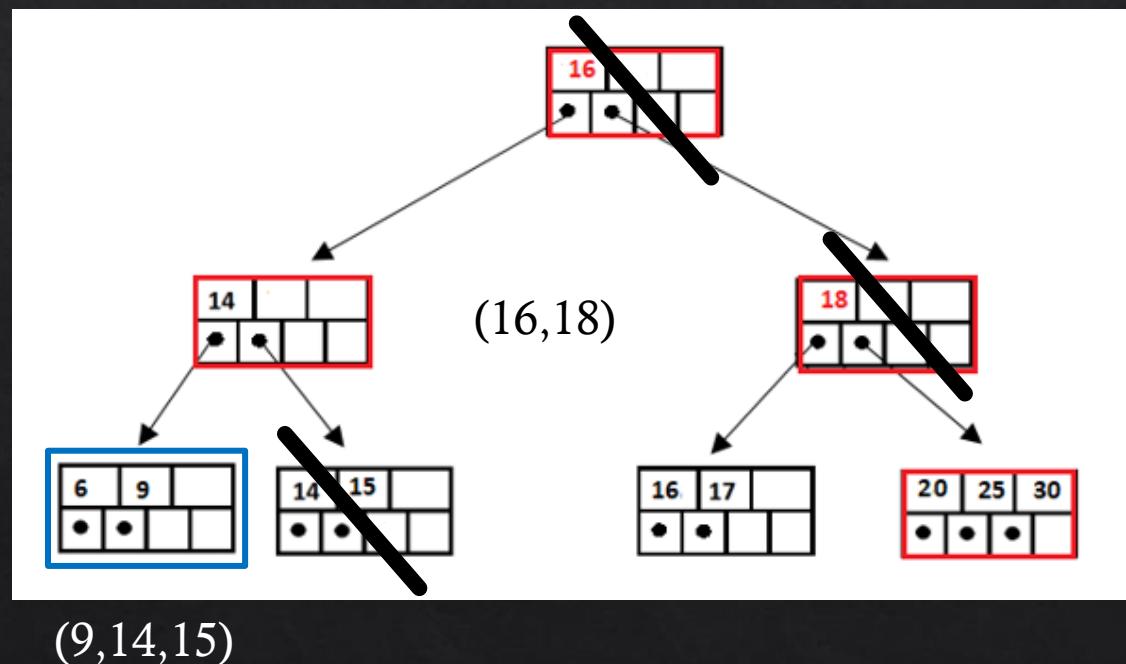
- ❖ A kulcs, ami elválasztotta a két részfát a szülőből is törlendő!
- ❖ 25 törlendő a szülőből is! Most a 25-ös belső csúcs lesz szabálytalan, mert csak egy gyereke maradt.
- ❖ Viszont testvére ki tudja segíteni: a (16,17) levelet átadja neki.
- ❖ Ennek kivitelezése:
Összerakjuk a megmaradt kulcsokat, a szülő kulcsát is bevonva: (14,16,18)
- ❖ A középső megy fel a szülőbe, a maradékon a két testvér megosztozik: ha nem lehet pontosan felezni, az kap több gyereket, akinek eredetileg is több volt.

21 törlése után kapott fa



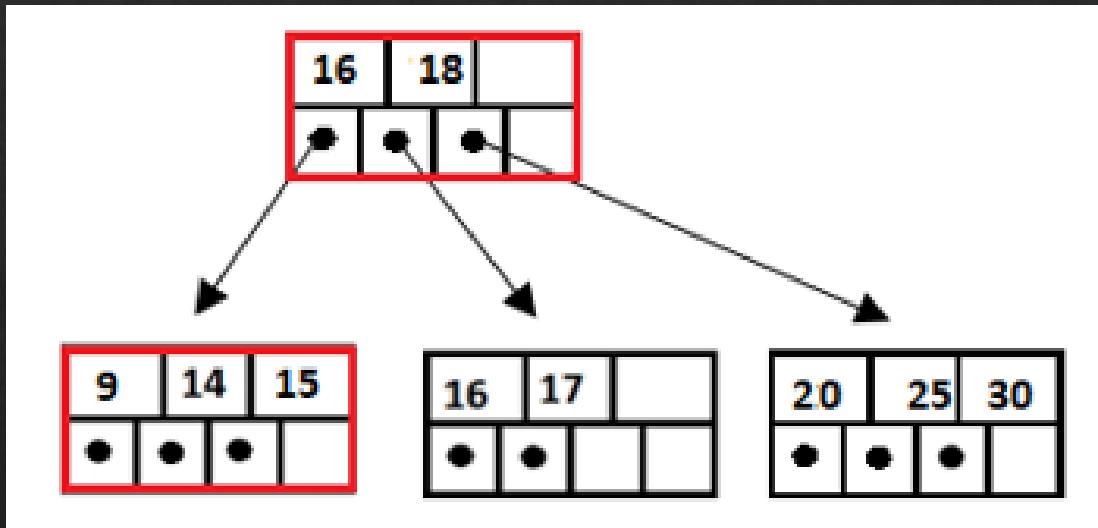
Az így kapott fából töröljük a 6-os kulcsot

- ❖ A levélben csak a 9-es maradna, a testvére nem tud átadni kulcsot, így jobb testvéréből átmásoljuk a 14 és 15 kulcsokat, majd a Node törlődik.
- ❖ A törlést meg kell ismételni a szülőre, de akkor a 14-es Node-nak csak egy gyereke marad.
- ❖ Nem tud a testvér sem átadni gyereket, így a két belső pont összevonódik.
- ❖ Belső pont összevonásakor egyesítik a megmaradt kulcsokat, a szülő kulcsát is bevonva: (16,18)
- ❖ A bal oldali Node marad meg, abból a 14 törölve lett, beleíródik a 16 és a 18, majd az eredeti 18-as Node törlődik.
- ❖ Ezután a törlő eljárás megismétlődik a szülőre (most a gyökérre): azt a kulcsot, mely a most egyesített testvéreket elválasztotta, ki kell törölni.
- ❖ Ekkor viszont a gyökérnek csak egy gyereke marad, így az eredeti gyökér törlődik, és a gyereke lesz az új gyökér.



6 törlése után kapott fa

- ◆ A fa magassága eggyel csökkent.



Gyakorló feladat

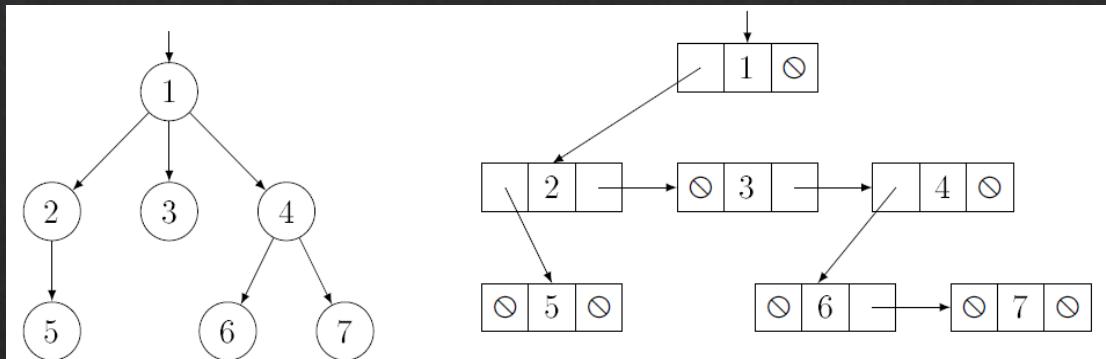
- ❖ Adott a következő B+ fa ($d=4$, azaz 4 pointer található minden csúcsban)
 $\{ [(2 \ 5) 9 (10 \ 11 \ 13) 16 (16 \ 17 \ 20) 21 (22 \ 23)] 27 [(29 \ 30 \ 32) 33 (34 \ 35) 40 (40 \ 42)] \}$
 - ❖ Rajzolja le, hogyan néz ki a fa.
 - ❖ Szűrja be a fába a 28 és 9 kulcsokat (mindig az eredeti fába szúrjon be).
 - ❖ Az eredeti fából törölje a 2 és 42 kulcsot (mindig az eredeti fából töröljön).

Rajzolja le a műveletek utáni állapotot – elég a fa gyökerét és a megváltozott részfát lerajzolni!

- ❖ Rajzolja fel az alábbi fát, majd törölje a 16-os kulcsot.
 $\{ [(2 \ 5) 12 (16 \ 17)] 22 [(22 \ 26 \ 30) 33 (35 \ 36) 38 (38 \ 40)] \}$

Általános fák

- ❖ Van egy kitüntetett csúcsa, a gyökér, a csúcsoknak tetszőleges sok leszármazottja lehet.
- ❖ Ábrázolás: két pointerrel, egyik az első leszármazottra mutat, a másik a testvére.
- ❖ Esetleg kiegészíthetjük szülő pointerrel is: a testvérek mindegyike a szülőjére mutat vissza.

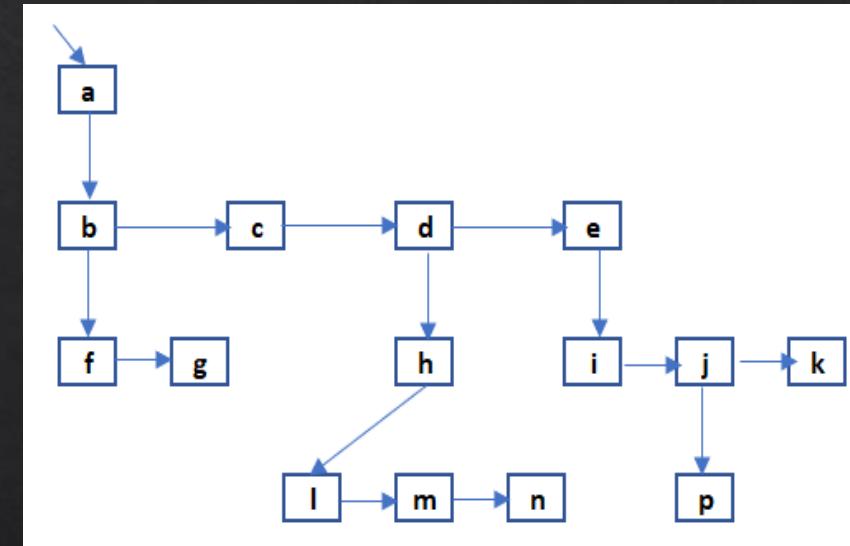
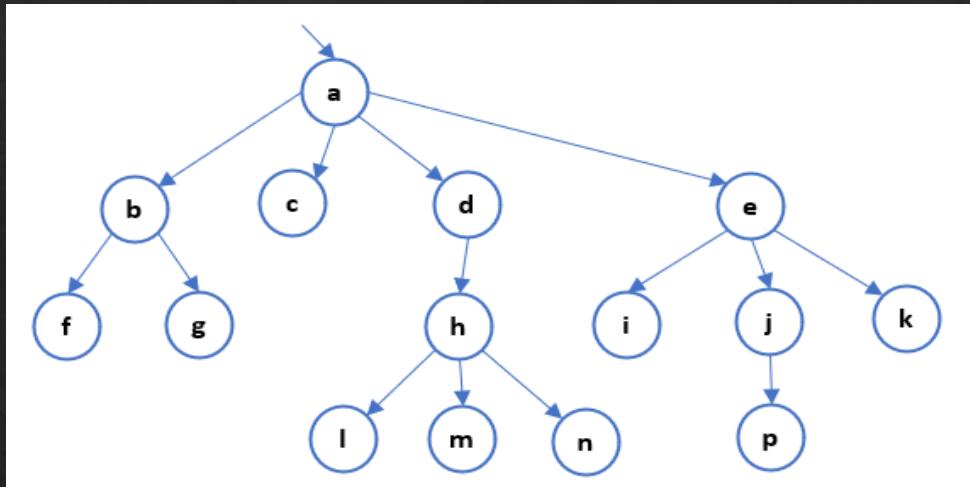


| Node |
|---|
| + <i>child1, sibling</i> : Node* // <i>child1</i> : első gyerek; <i>sibling</i> : következő testvér |
| + <i>key</i> : T // T ismert típus |
| + Node() { <i>child1</i> := <i>sibling</i> := ⊖ } // egyszúcsú fát képez belőle |
| + Node(<i>x</i> :T) { <i>child1</i> := <i>sibling</i> := ⊖ ; <i>key</i> := <i>x</i> } |

Megjegyzés: Másfélé ábrázolás is lehetséges, például a gráfok ábrázolásához használatos éllistás módszert is használhatjuk.

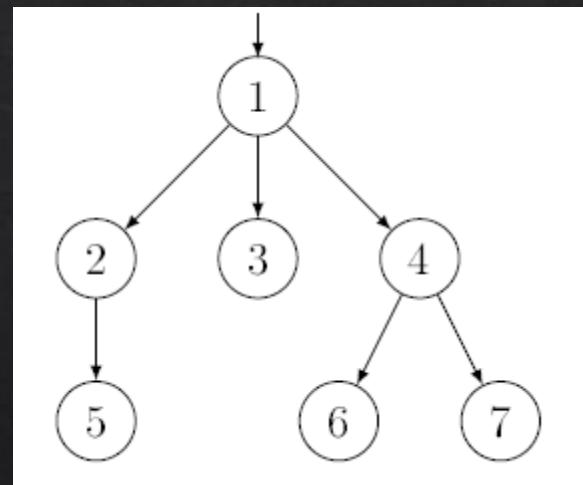
Feladat

- ❖ Adott a képen látható általános fa, rajzoljuk le két pointeres ábrázolásban:



Zárójelezett alak

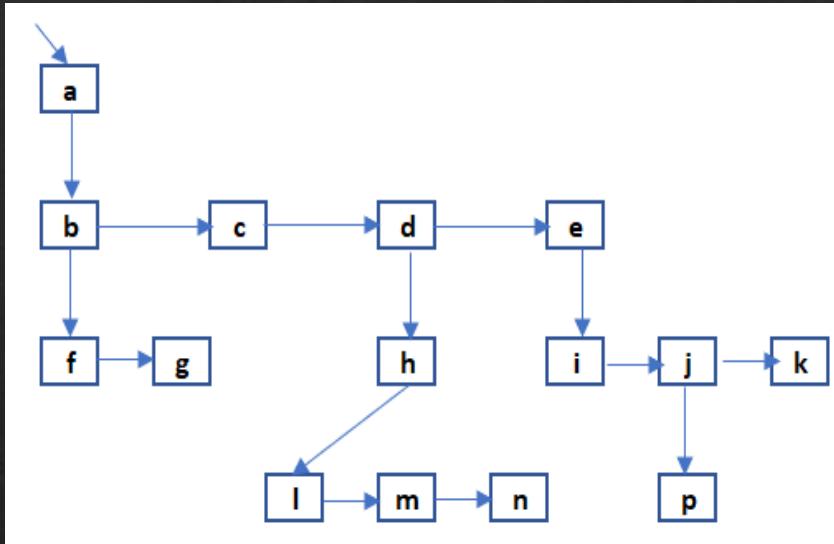
- ❖ Általános fákat leírhatjuk zárójelezéssel.
- ❖ A fa szöveges leírása.
- ❖ Egy nemüres fa általános alakja ($G \ t_1 \dots t_n$), ahol G a gyökérkötél tartalma, $t_1 \dots t_n$ pedig a részfák.
- ❖ Így például a képen látható fa zárójelezett leírása a következő: $\{ 1 [2 (5)] (3) [4 (6) (7)] \}$
- ❖ Megjegyzés: nem szükséges a zárójel típusokat váltogatni, csak a könnyebb olvashatóságot szolgálja. A fenti leírásban a levelek () zárójelben vannak.



Feladat

- ❖ Készítsük el a példaként megadott fa zárójeles leírását!

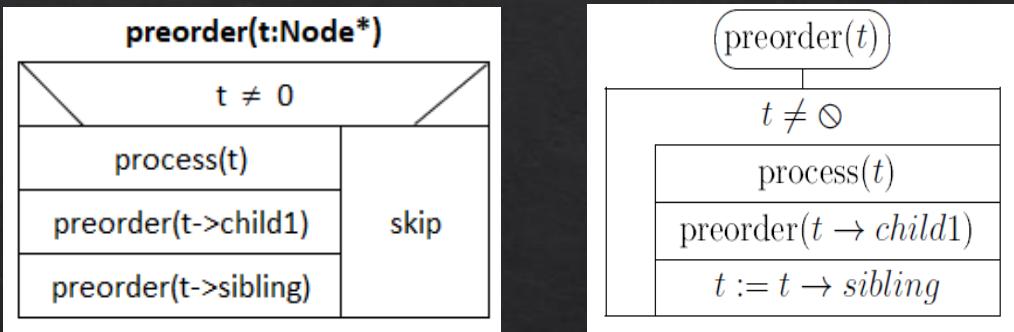
- ❖ Váltogassuk a zárójelek típusát szisztematikusan:
 $\{ \} [] ()$



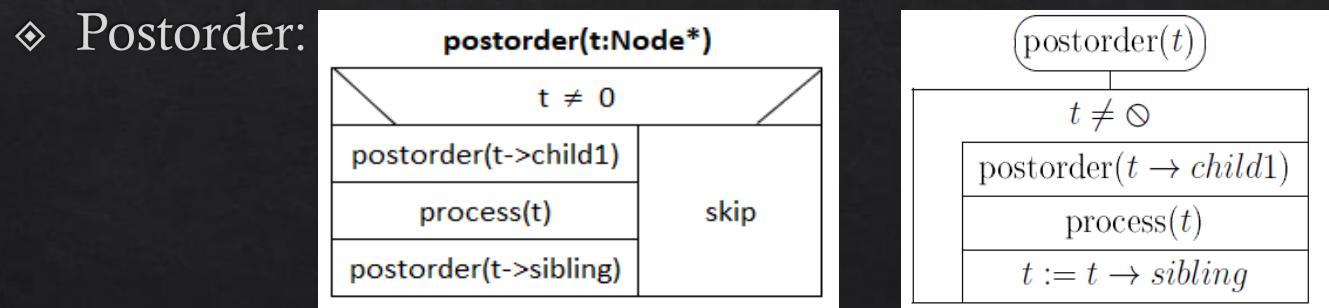
- ❖ $\{ a [b (f)(g)] [c] [d (h \{ l \} \{ m \} \{ n \})] [e (i)(j \{ p \})(k)] \}$

Bejárások

- ❖ A bináris fákra tanult rekurzív bejárások általánosíthatók a két pointerrel ábrázolt általános fákra. Ezen az ötleten alapszik a két következő bejárás:
- ❖ Preorder:



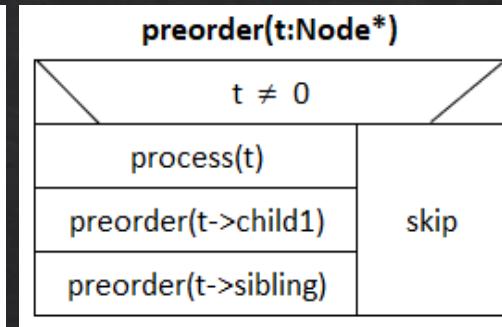
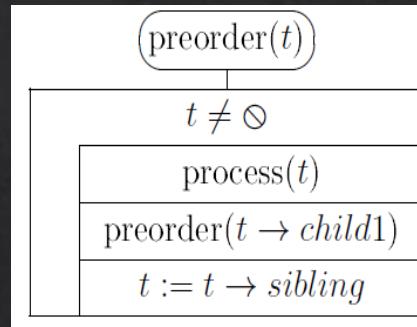
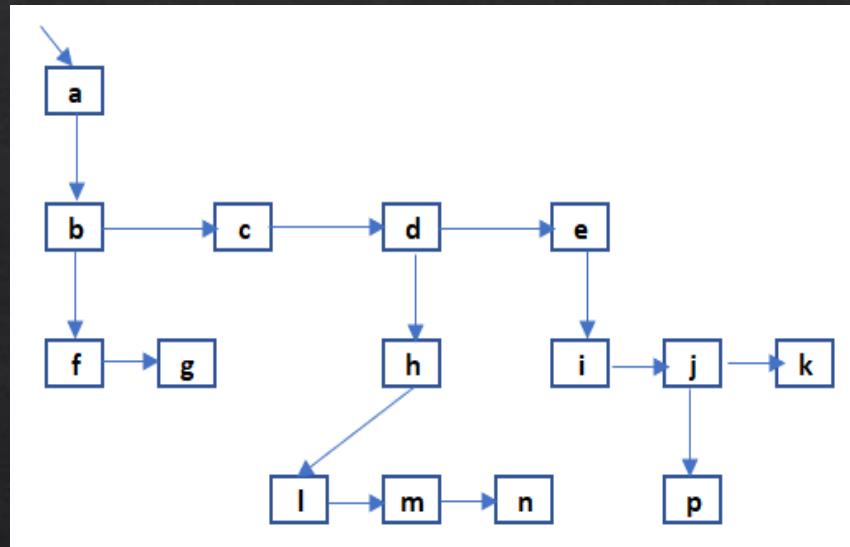
Megjegyzés: a jegyzetben található algoritmus a testvérek irányában ciklust használ, így hatékonyabb.



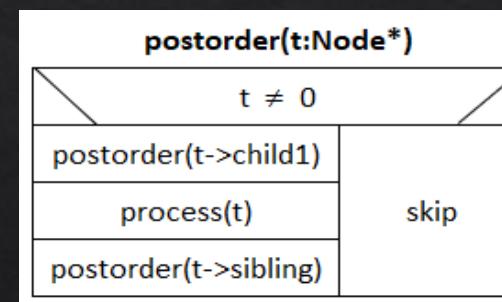
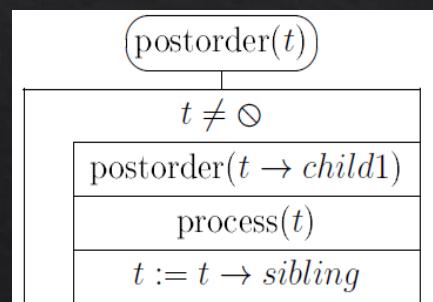
Megjegyzés: alakilag ez inkább a bináris fák inorder bejárására emlékeztet, hogy miért ezt definiáljuk postorderként, azt egy későbbi példán bemutatjuk.

Feladat

- ❖ Kövessük végig a bejárásokat a példa fán!



a | b | f | g | c | d | h | l | m | n | e | i | j | p | k

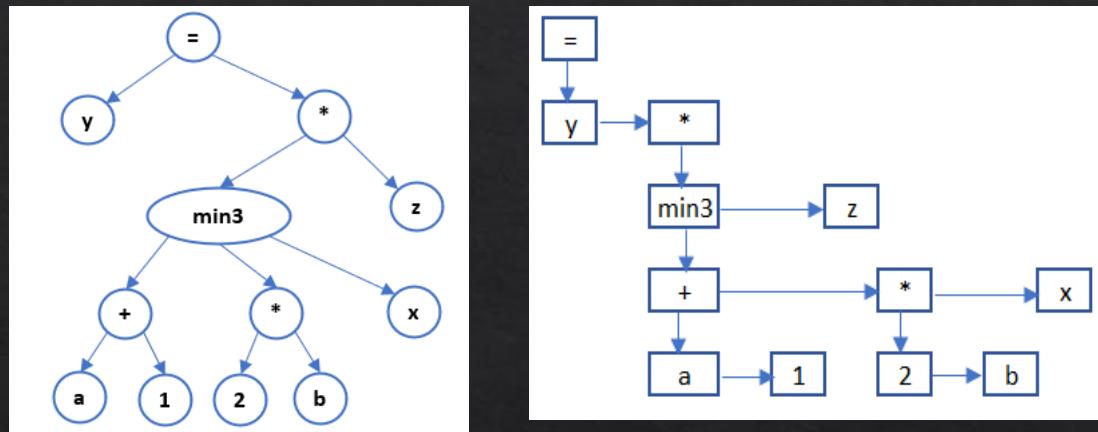


f | g | b | c | l | m | n | h | d | i | p | j | k | e | a

| postorder(t:Node*) | |
|-----------------------|------|
| t ≠ 0 | |
| postorder(t->child1) | skip |
| process(t) | |
| postorder(t->sibling) | |

Miért ezt az alakot definiáljuk postorder bejárásként?

- Tekintsük a következő függvény kifejezést: $y = \min3(a+1, 2*b, x) * z$
 $\min3$ legyen egy három paraméteres függvény.
- Ábrázoljuk ezt egy általános fával!



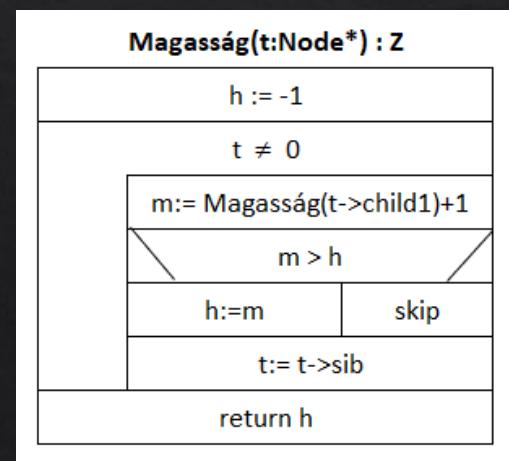
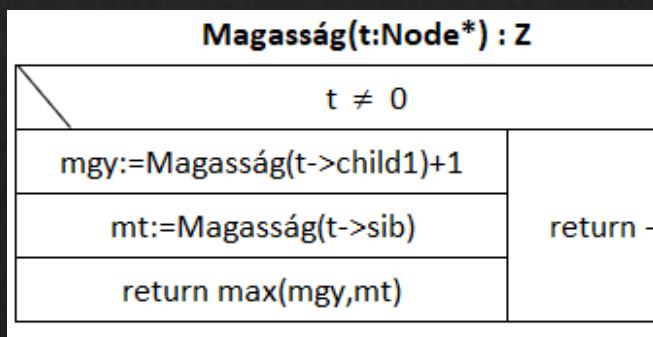
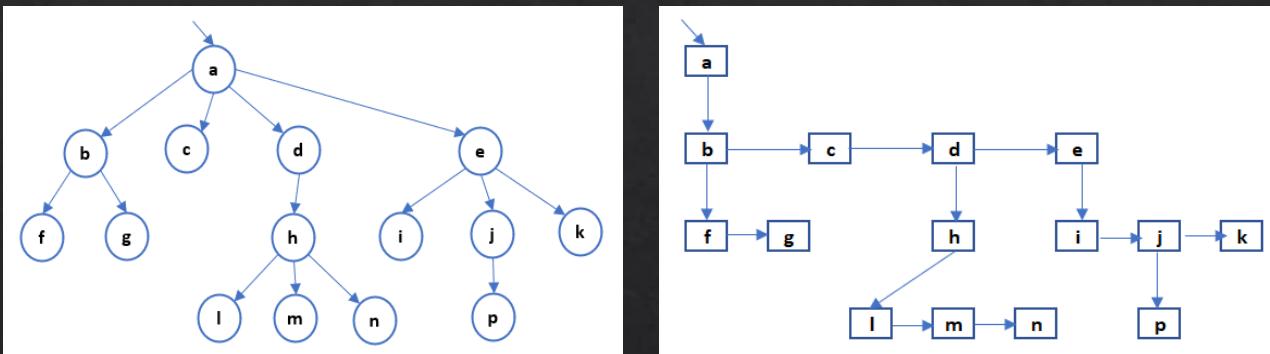
Melyik bejárás adta bináris fák esetén a kifejezés lengyel formáját?

- Próbáljuk ki a postorder bejárást, vajon megkapjuk a lengyel formát?



Magasság

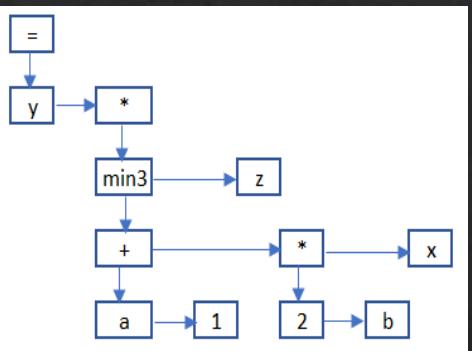
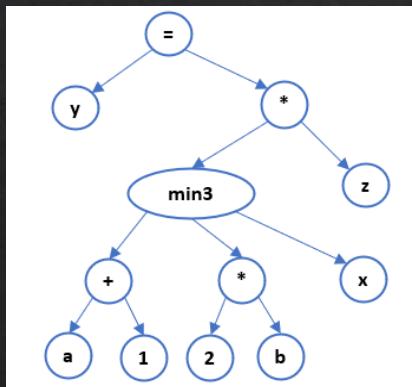
- ❖ Készítsünk (rekurzív) algoritmust, mely meghatározza egy két pointerrel ábrázolt általános fa magasságát.
- ❖ A példa fa magassága: 3



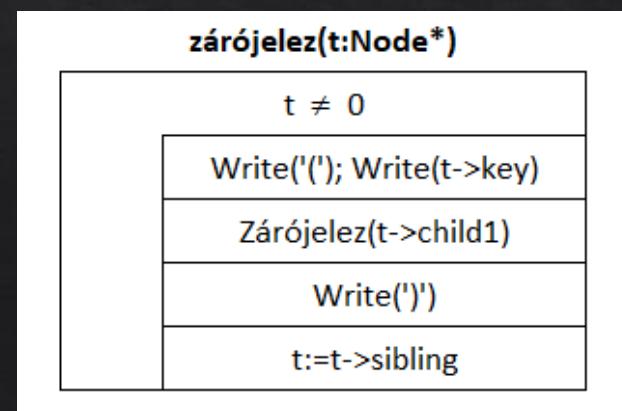
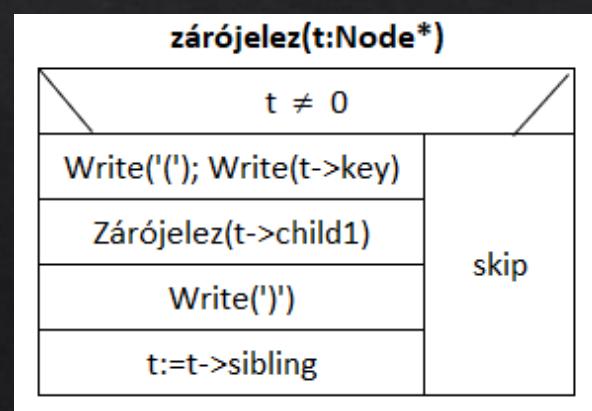
A testvérek irányában ciklust használó algoritmus hatékonyabb, ezért előnyben részesítjük.

Zárójelezett alak

- ❖ Készítsünk algoritmust, mely kiírja a fa zárójelezett alakját. Használunk csak () zárójeleket.

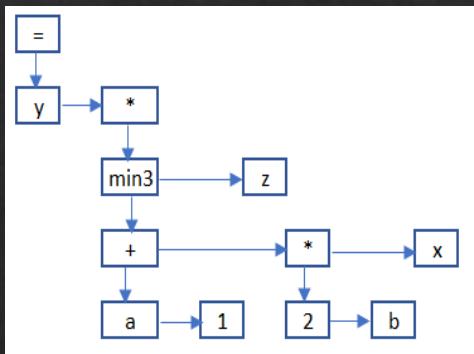


(= (y) (* (min3 (+ (a) (1)) (* (2) (b))) (x)) (z)))



Zárójelezett alak (gyakorló feladat)

- ◆ Fejlesszük tovább az algoritmust! Használunk szisztematikusan 3 féle zárójel típust:
 $\{ \}$ $[]$ $()$

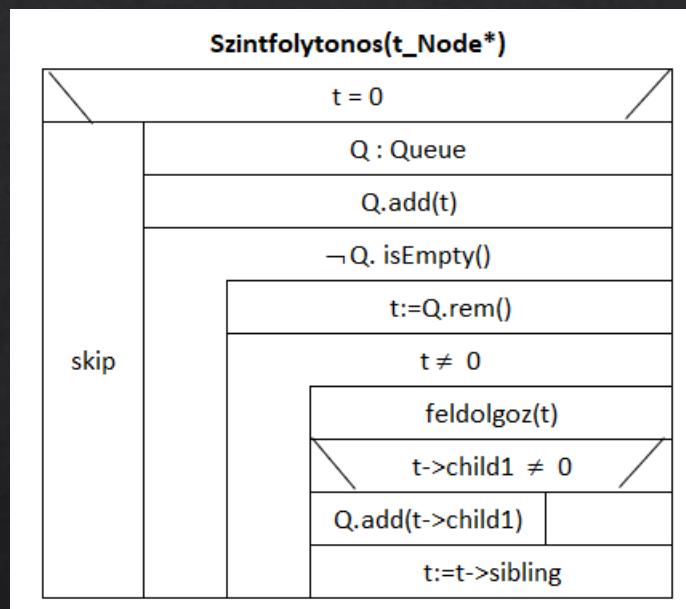
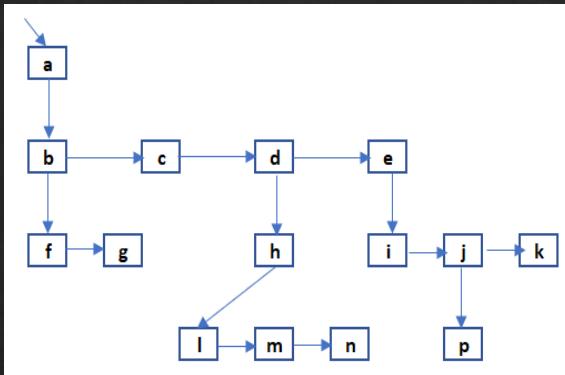


$= [y] [* (min3 {+ [a] [1]} {* [2] [b]} {x}) (z)]$

- ◆ Vegyük észre, hogy a zárójel típust az határozza meg, melyik szinten vagyunk a fában:
 - ◆ 0, 3, 6, ... $\{ \}$
 - ◆ 1, 4, 7, ... $[]$
 - ◆ 2, 5, 8, ... $()$
- ◆ Ötlet: egy paramétert beteszünk még az algoritmusba, mely azt adja meg, milyen szinten járunk a fában, és a szinttől függően választjuk a megfelelő zárójelt.

Szintfolytonos bejárás

- Bináris fáknál tanultuk a szintfolytonos bejárást, készítsük el két pointerrel ábrázolt általános fákra is az algoritmust!



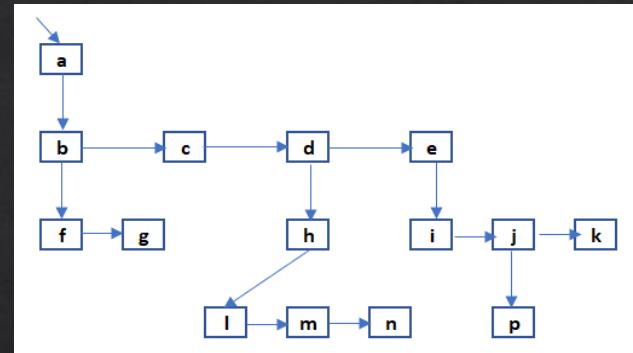
| Feldolgozott csúcs | Sor tartalma |
|--------------------|--------------|
| a | a |
| a | b |
| b | f |
| c | f |
| d | f h |
| e | f h i |
| f,g | h i |
| h | i l |
| i | l |
| j | l p |
| k | l p |

Szorgalmi hf

- ❖ Mutassunk az órán bemutatott példák alapján olyan eseteket, amikor egy alapból legalább 2 magas fa szintje
 - ❖ eggel nő
 - ❖ eggel csökken
 - ❖ változatlan marad, de azért valami érdekes történik. ☺

Szorgalmi hf

- ❖ Adott egy két pointerrel, láncoltan ábrázolt általános fa. Egy csúcsban az első gyerekre és a testvérré mutató pointerek vannak. Készítse el a következő kereső algoritmust: megadunk egy kulcsot, ha a fában van ilyen kulcs, akkor kiírja a csúcsig vezető úton a szülők kulcsait, Elképzelhető, hogy az adott csúcs több helyen is szerepel a fában, akkor mindegyik előforduláshoz írja ki az útvonalat! Az utat a gyökértől indulva, a csúcsig kell kiírni, csak a szülő Node-ok kulcsait.
- ❖ Érdekesség képpen oldjuk meg úgy is, hogy a Node-okban van egy szülő (parent) pointer is. Ha megtaláljuk a keresett kulcsot egy Node-ban, akkor a parent pointerek segítségével írjuk ki az útvonalat.
- ❖ *Megjegyzés: a feladat valós probléma. Például amikor az operációs rendszerek könyvtár rendszerében keresünk egy adott fájl bejegyzést: hol fordul elő egy adott nevű fájl? Lehet több alkönyvtárban is ugyanolyan nevű fájl. Írjuk ki az elérési útvonalakat.*



Keresett kulcs: „n”
Elérési út: a/d/h

Keresett kulcs: „g”
Elérési út: a/b

Keresett kulcs: „a”
Elérési út: (üres szöveg)