



Pázmány Péter Katolikus Egyetem
Információs Technológiai és Bionikai Kar

Külső rendezők 2-3 fák, B-fák

12. előadás



Pázmány Péter Katolikus Egyetem
Információs Technológiai és Bionikai Kar

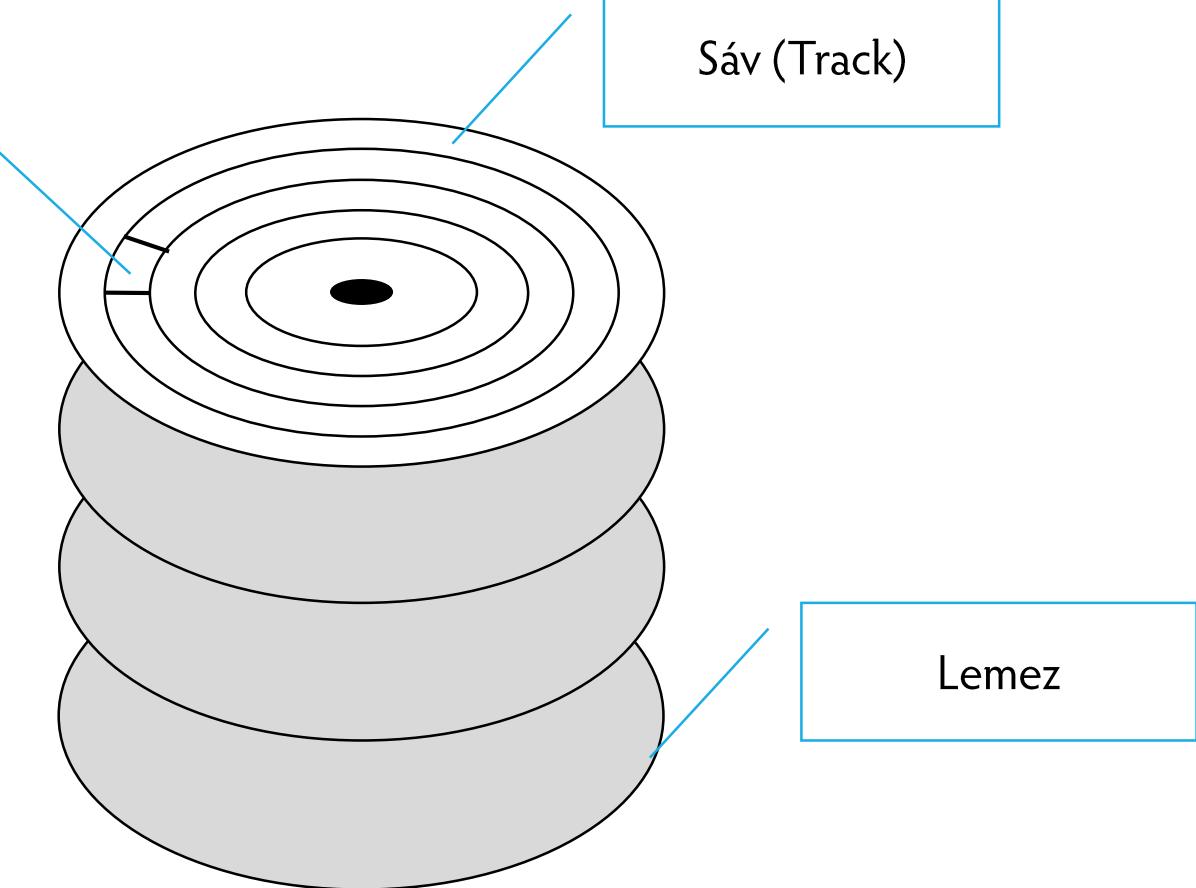
Külső rendezések



Adattárolás séma

- Az adattárolás egy háttértárolón (lemezen) nem bit/byte egységbe van szervezve
 - Nagyobb egységek: lap (HDD: szektor, SSD: blokk)
 - Ez lehet például 2048 byte, vagy 4096 byte
 - Ez az átvitel egysége!
 - Lényegében az átvitelek száma határozza meg a sebességet
 - A háttértárról egy lap olvasása lassabb, mint a főmemória olvasása
 - ~ms vs. ~ns
 - Még SSD esetén is!

Mágneslemez esetén





Külső rendezések

- Az eddig látott rendezésekben feltettük, hogy az adatok a központi memoriában vannak
 - Ennek megfelelt, hogy a hatékonyságot az összehasonlítások számában mértük.
- Ha az adatok háttértárban vannak, akkor a futási idő döntő részét az I/O utasítások teszik ki.
 - Az I/O egysége az 1 blokk, ami $k * 512$ byte valamely kis k -val, pl. 1024 v. 2048 byte.
 - A hatékonyságot a szükséges **blokk I/O**-k számában mérjük.
- Külső rendezésre igazából csak az összefésüléses rendezés (MergeSort) alkalmas.
 - Miért?



Külső rendezések

- Az összefésüléses rendezés külső tárakon
 - Adott egy S szekvenciális input fájl, amely n blokkból áll, minden blokkban adott számú rekorddal.
 - Pl. 1 blokk = 1024 byte és ezen 4 rekord foglal helyet.
 - A blokkok tartalma rendezetlen.
 - Az összefésülést iteratív módon végezzük, úgy, hogy az egyes „menetek” végén egyre nagyobb darabok, vagyis egyre több szomszédos blokk lesz rendezett.
 - Az összefésülést menetenként váltakozva az A, B, illetve a C,D fájlokba végezzük, végül a teljesen rendezett eredményt S-be írjuk. A közbülső menetekben azért lesz 2 output fájl, mert az összefuttatás eredményét az „egyet ide, egyet oda” elv alapján írjuk ki.

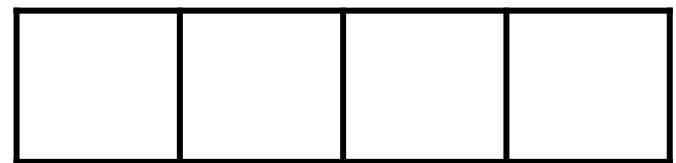
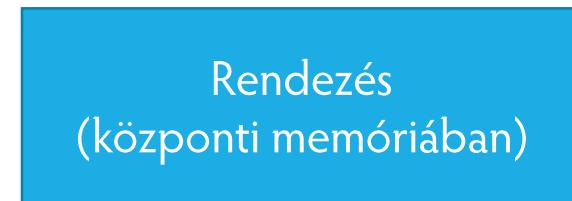
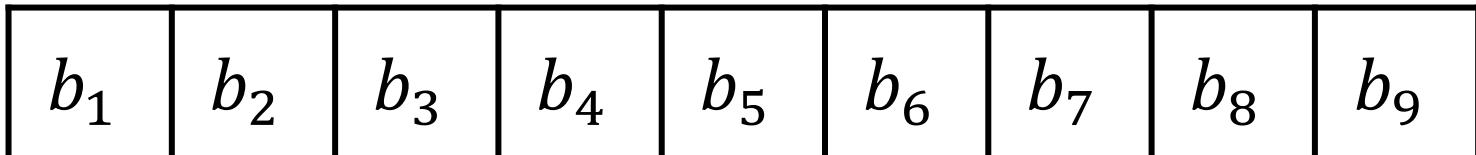


Külső rendezések

- 1. menet
 - Beolvassuk S rendezetlen blokkjait, valamilyen belső rendezővel rendezzük, majd kiírjuk felváltva A-ba, illetve B-be.
 - Itt még nem volt összefésülés.

Külső rendezések

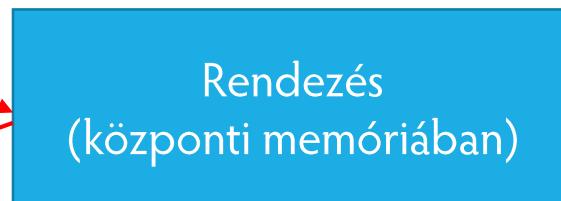
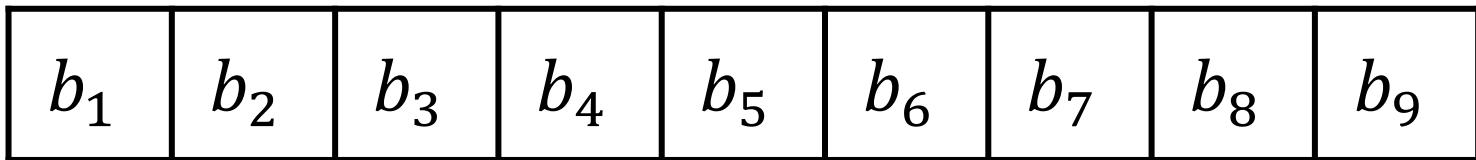
- S: rendezetlen blokkok



Önmagában rendezett blokkokat tartalmaznak

Külső rendezések

- S: rendezetlen blokkok



A

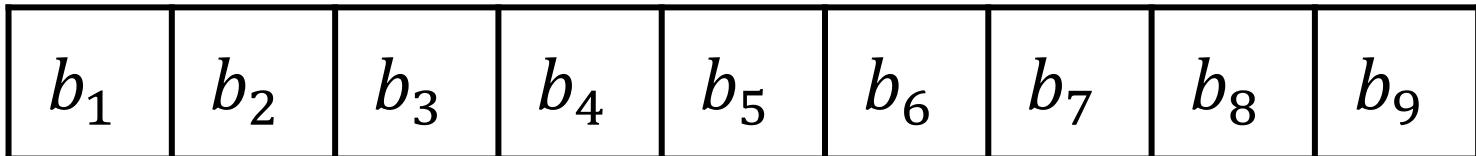
B



Önmagában rendezett blokkokat tartalmaznak

Külső rendezések

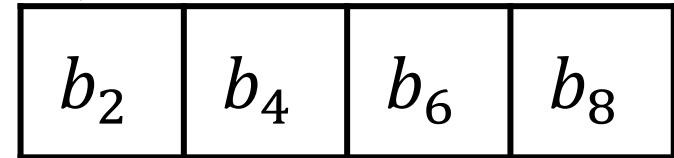
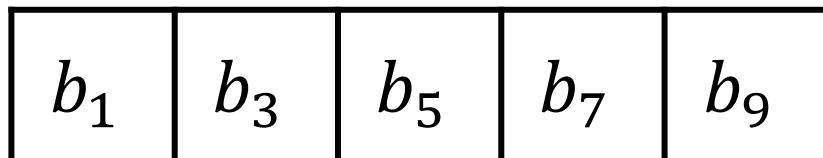
- S: rendezetlen blokkok



A

Rendezés
(központi memóriában)

B



Önmagában rendezett blokkokat tartalmaznak



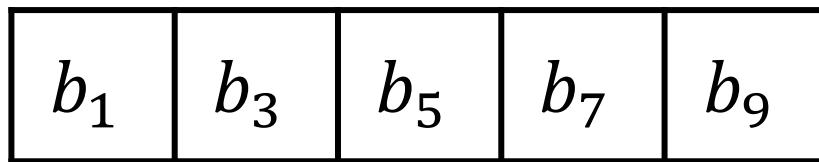
Külső rendezések

- 2. menet
 - Sorban beolvassuk A és B 1-1 blokkját. Ezek rendezettek. Összefésüljük őket és a rendezett két blokk hosszú adatot felváltva C-be, illetve D-be írjuk.
 - Az A utolsó blokkjának nincs párja, így azt kiírjuk C-be. A C és D fájlban a rendezett részek hossza 2 blokk, illetve a maradék esetében 1 blokk.

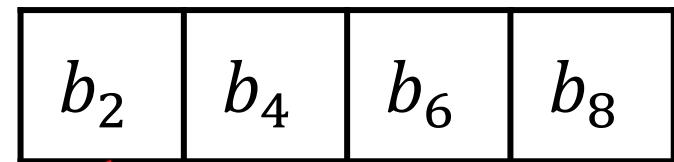
Külső rendezések

A

Rendezett: 1 blokk



B

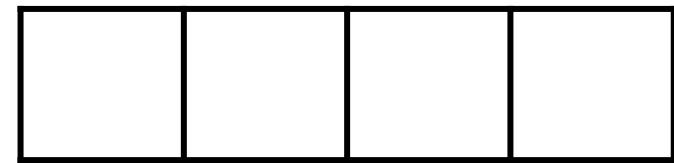


C

Összefuttatás
(RAM)



D



Rendezett: 2 blokk

Külső rendezések

A

Rendezett: 1 blokk

b_1	b_3	b_5	b_7	b_9
-------	-------	-------	-------	-------

B

b_2	b_4	b_6	b_8
-------	-------	-------	-------

C

Összefuttatás
(RAM)

b_1^-	$-b_2$			
---------	--------	--	--	--

D

b_3^-	$-b_4$		
---------	--------	--	--

Rendezett: 2 blokk

Külső rendezések

A

Rendezett: 1 blokk

b_1	b_3	b_5	b_7	b_9
-------	-------	-------	-------	-------

B

b_2	b_4	b_6	b_8
-------	-------	-------	-------

C

Összefuttatás
(RAM)

b_1^-	$-b_2$	b_5^-	$-b_6$	b_9
---------	--------	---------	--------	-------

D

b_3^-	$-b_4$	b_7^-	$-b_8$
---------	--------	---------	--------

Rendezett: 2 blokk



Külső rendezések

- 3. menet
 - C-ből és D-ből olvasunk 2-2 rendezett blokkot, összefésüljük őket és felváltva A-ba és B-be írjuk a rendezett 4 blokkot
 - A C-beli utolsó töredék blokk változatlanul A végére kerül

Külső rendezések

A

Rendezett: 4 blokk

b_1^-	b_2	b_3	$-b_4$	
---------	-------	-------	--------	--

B

--	--	--	--

C

Összefuttatás
(RAM)

b_1^-	$-b_2$	b_5^-	$-b_6$	b_9
---------	--------	---------	--------	-------

D

b_3^-	$-b_4$	b_7^-	$-b_8$
---------	--------	---------	--------

Rendezett: 2 blokk

Külső rendezések

A

Rendezett: 4 blokk

b_1^-	b_2	b_3	$-b_4$	
---------	-------	-------	--------	--

B

b_5^-	b_6	b_7	$-b_8$
---------	-------	-------	--------

C

Összefuttatás
(RAM)

b_1^-	$-b_2$	b_5^-	$-b_6$	b_9
---------	--------	---------	--------	-------

D

b_3^-	$-b_4$	b_7^-	$-b_8$
---------	--------	---------	--------

Rendezett: 2 blokk

Külső rendezések

A

Rendezett: 4 blokk

b_1^-	b_2	b_3	$-b_4$	b_9
---------	-------	-------	--------	-------

B

b_5^-	b_6	b_7	$-b_8$
---------	-------	-------	--------

C

Összefuttatás
(RAM)

b_1^-	$-b_2$	b_5^-	$-b_6$	b_9
---------	--------	---------	--------	-------

D

b_3^-	$-b_4$	b_7^-	$-b_8$
---------	--------	---------	--------

Rendezett: 2 blokk



Külső rendezések

- 4. menet
 - C-be kerül A és B 4-4 rendezett blokkja összefésülésének a 8 blokk hosszú rendezett eredménye, D-be pedig a maradék egy blokk.

Külső rendezések

A

Rendezett: 4 blokk

b_1-	b_2	b_3	$-b_4$	b_9
--------	-------	-------	--------	-------

B

b_5-		b_7	$-b_8$
--------	--	-------	--------

C

Összefuttatás
(RAM)

b_1-	b_2	b_3	b_4	b_5	b_6	b_7	$-b_8$
--------	-------	-------	-------	-------	-------	-------	--------

D

b_9

Rendezett: 8 blokk



Külső rendezések

- 5. menet
 - C 8 blokkját és D 1 blokkját összefésüljük S-be.
 - Elnevezés: egy k blokkból álló összefüggő rendezett részt **k hosszú futam**-nak nevezünk.

Külső rendezések

S – Rendezett

b_1^-	b_2	b_3	b_4	b_5	b_6	b_7	b_8	$-b_9$
---------	-------	-------	-------	-------	-------	-------	-------	--------

Összefuttatás
(RAM)

C

D

b_1^-	b_2	b_3	b_4	b_5	b_6	b_7	$-b_8$	
---------	-------	-------	-------	-------	-------	-------	--------	--

b_9

Rendezett: 8 blokk



Külső rendezések

- Két kiegészítő megjegyzés:

1. Ebben a példában az S 9. blokkja csaknem végig nem került kapcsolatba más rendezett részekkel, csak az utolsó menetben került összefésülésre.
 - Ha végrehajtjuk a fenti eljárást $n = 15$ -re, akkor a páratlan töredék maradék rész mérete így alakul: 1, 3, 7, vagyis a fenti jelenség nem törvényszerű.
2. Ha a központi memória mérete korlátozott és nem képes befogadni az egyre növekvő méretű rendezett részeket (futamokat), akkor ezek összefésülését lehet „pufferelve”, akár blokkonként végezni.
 - Ez azért lehetséges, mert az összefuttatás egysége a rekord.



Külső rendezések

- Általában
 - 1. menet eredménye: 1 hosszú futamok
 - 2. menet eredménye: 2 hosszú futamok
 - 3. menet eredménye: 4 hosszú futamok
 -
 - $(k - 1)$. menet eredménye: 2^{k-2} hosszú futamok
 - k . (utolsó) menet eredménye: $\leq 2^{k-1}$ hosszú egyetlen futam = S
 - előtte maradék mindenkor lehet



Külső rendezések

- Az utolsó előtti $(k - 1)$. menetben még volt 2 futam
 - $2^{k-2} < n$
- A k . (utolsó) menetben már előáll a teljes rendezett fájl
 - $n \leq 2^{k-1}$
- Áttérve logaritmusra:
 - $k - 2 < \log_2 n \leq k - 1 \Rightarrow k - 1 = \lceil \log_2 n \rceil$
 - A menetek k száma: $k = \lceil \log_2 n \rceil + 1$
 - Az összes blokk-I/O száma: $2 * n * (\lceil \log_2 n \rceil + 1)$
 - minden menetben beolvastuk és kiírtuk mind az n blokkot



Külső rendezések

- Gyorsítási lehetőségek:
 - Nagyobb kezdő futamok létrehozása
 - Ha a központi memória lehetővé teszi, akkor az 1. menetben megtehetjük, hogy S-ből $m > 1$ blokkot olvasunk be, ezt rendezzük és az így keletkezett m hosszú kezdőfutamokat írjuk ki A-ba és B-be.
 - Ezután úgy megy tovább, hogy először két m hosszú futamokat fésülünk össze, majd két $2m$ hosszút, stb.



Külső rendezések

- Számítások
 - 1. menet eredménye: m hosszú futamok
 - 2. menet eredménye: $2 * m$ hosszú futamok
 - 3. menet eredménye: $4 * m$ hosszú futamok
 -
 - $(k - 1)$. menet eredménye: $2^{k-2} * m$ hosszú futamok
 - k . (utolsó) menet eredménye: $\leq 2^{k-1} * m$ hosszú egyetlen futam = S
 - Innen $2^{k-2} * m < n \leq 2^{k-1} * m$
 - A menetek k száma: $k = \left\lceil \log_2 \frac{n}{m} \right\rceil + 1$
 - Az összes blokk-I/O száma: $2 * n * \left(\left\lceil \log_2 \frac{n}{m} \right\rceil + 1 \right)$

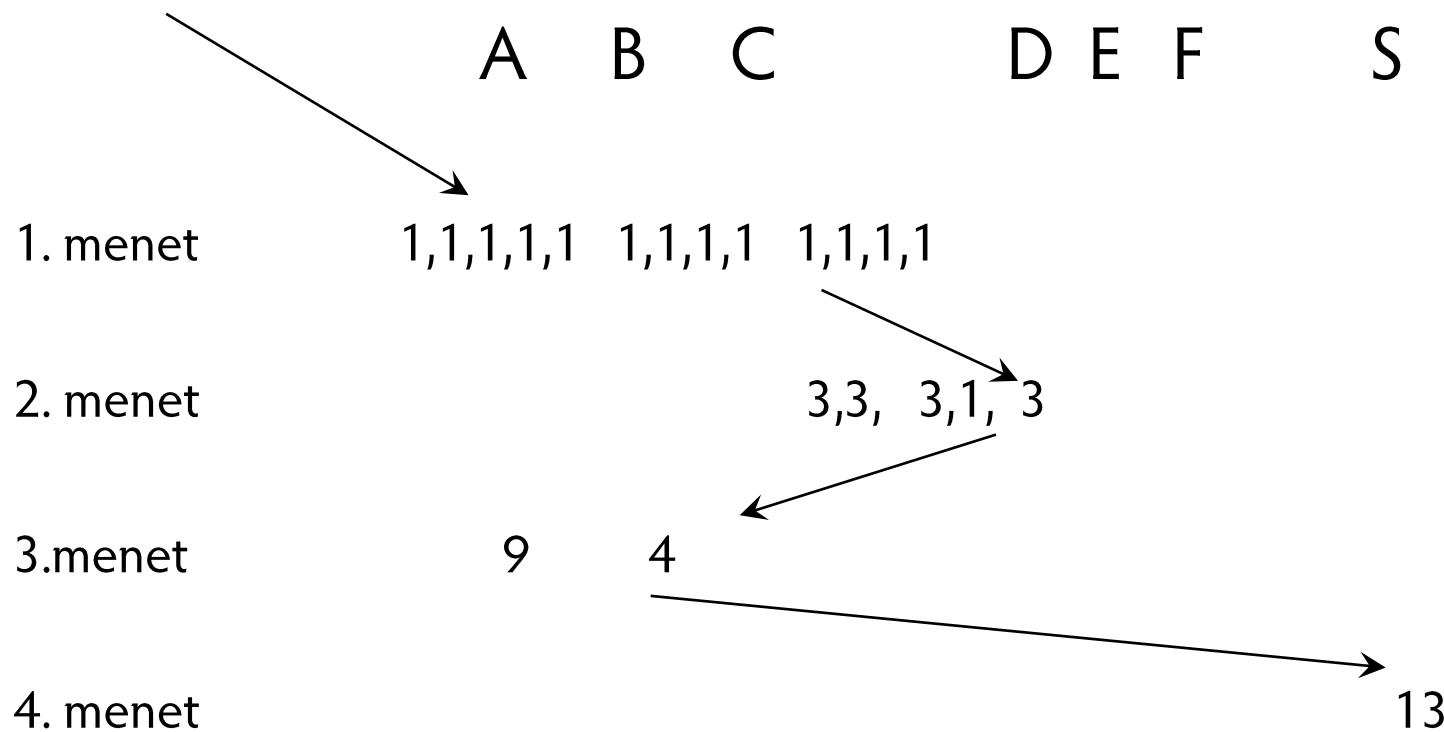


Külső rendezések

- Lehet azt is, hogy több, mint kétfelé fésülünk:
 - Ha az S fájl mellett nem $2 * 2$, hanem általában $2 * m$ fájllal dolgozunk, akkor ezzel a ráfordítással hatékonyabb eljárás nyerhető.
 - Legyen például $m = 3, n = 13$ és térjünk vissza az 1 hosszú kezdő futamokhoz:
 - Ekkor az 1. menetben felváltva A, B, C-be írjuk ki a rendezett kezdőfutamokat.
 - Ezután minden három futamot fésülünk össze (a maradékoktól eltekintve) és az új futamokat felváltva A, B, C-be ill. D, E, F-be írjuk ki.
 - Az utolsó menetben S-be írjuk az eredményt.

Külső rendezések

S-ben n= 13





Külső rendezések

- A számolás

- 1. menet eredménye: 1 hosszú futamok
- 2. menet eredménye: m hosszú futamok
-
- $(k - 1)$ -edik menet eredménye: m^{k-2} hosszú futamok
- k -adik utolsó menet eredménye: $\leq m^{k-1}$ hosszú egyetlen futam=S
- Innen: $m^{k-2} < n$

$$k = \lceil \log_m n \rceil + 1 = \left\lceil \frac{\log_2 n}{\log_2 m} \right\rceil + 1$$

$$m^{k-1} \geq n$$

- Az összes blokk I/O művelet: $2n * \left(\left\lceil \frac{\log_2 n}{\log_2 m} \right\rceil + 1 \right)$



Külső rendezések

- Három fájlos külső rendező

- Érdekes algoritmushoz jutunk, ha az m -felé fésülésnél nem $2m$ db fájl-t használunk, hanem csak $m + 1$ db-t
- Ezt $m = 2$ esetére nézzük meg, ami az eredeti eset. Ekkor tehát 3 fájl-t használunk: A, B, C-t
- Az első menetben szétosztjuk S immár rendezett blokkjait A-ba és B-be.
- A második menetben elkezdjük A és B blokkjainak az összefésülését, de most csak 1 fájl tudja fogadni az eredményt, a C fájl. Ezért C-be fésülünk össze egészen addig, amíg A és B egyike ki nem ürül
- Ekkor új menetet kezdünk a két nem üres fájllal
 - És így tovább



Külső rendezések

- Példa: $S n = 13$ blokk

- A táblázatban az első szám a futamok számát jelenti, zárójelben pedig a futamok hossza áll

	1.	2.	3.	4.	5.	6.	7.	8.
A	7(1)	1(1)	0	1(5)	0	1(9)	0	1(13)
B	6(1)	0	1(3)	0	1(7)	0	1(11)	0
C		6(2)	5(2)	4(2)	3(2)	2(2)	1(2)	0

- Látjuk, hogy a második menet végén képződött 6 db 2 hosszú futam csak egyesével tud elfogyni – érezhetően sok lépésben
- Ennek oka az, hogy az 1. menet végi két futamszámnak 1 a különbsége
 - $7 - 6 = 1$
 - Hogy tudnánk ezen javítani?



Külső rendezések

- Eszünkbe jut a Fibonacci sorozat: 0,1,1,2,3,5,8,13,...
 - Ahol két szomszédos elem különbsége – az első néhány tag kivételével – 1-től különböző
 - Innen jön a gondolat, hogy az első menetben írunk annyi futamot A-ba és B-be, mint a Fibonacci sorozat két (alkalmas) szomszédos eleme, és lépkedjünk visszafelé a sorozaton az egyes menetekben

	1.	2.	3.	4.	5.	6.
A	8(1)	3(1)	0	2(5)	1(5)	0
B	5(1)	0	3(3)	1(3)	0	1(13)
C		5(2)	2(2)	0	1(8)	0



Külső rendezések

- **Belátható**, hogy a 3 fágos rendező éppen akkor fut le leggyorsabban, ha így járunk el, azaz A-ban és B-ben két szomszédos Fibonacci számnak megfelelő számú kezdő futamot hozunk létre.
 - Ha N nem Fibonacci szám, akkor vagy levágjuk és félretesszük a felesleget, és a végén még összefésüljük a kialakult eredménnyel, vagy pedig éppen fordítva: (virtuális) kiegészítéssel alkalmasan megnöveljük az input állomány méretét.



Külső rendezések

- Nézzük meg, hány lépében érjük el F_k és F_{k-1} -ből az 1, 0 számokat úgy, hogy minden menetben egy újabb számot tudunk lefelé lépni

	1.	2.	...	$(k - 1)$.	$k.$
$ S \approx F_{k+1}$	F_k	F_{k-1}	...	1	1
	F_{k-1}	F_{k-2}	...	1	0

- Látható, hogy a menetek száma ekkor k, hiszen F_k -től F_1 -ig vezet az út a táblázatban.
- Ki kell fejezni n -et F_{k+1} -gyel, ami az input fájl (közelítő) mérete blokkokban. Jelöljük $N = F_{k+1}$



Külső rendezések

- Számítás menete

- $N = F_{k+1}$
- $F_0 = 0, F_1 = 1, (F_2 = 1, F_3 = 2, F_4 = 3, \dots)$ $F_{k+1} = F_k + F_{k-1} (k \geq 1)$
- $F_k = \frac{1}{\sqrt{5}} * \left[\left(\frac{(1+\sqrt{5})}{2} \right)^k - \left(\frac{(1-\sqrt{5})}{2} \right)^k \right]$ $\Rightarrow F_k \approx \frac{1}{\sqrt{5}} * \left(\frac{(1+\sqrt{5})}{2} \right)^k$
$$\frac{1,6180}{2} - \frac{0,6180}{2}$$
- Az $A = \frac{(1+\sqrt{2})}{2}$ az aranymetszés aránya, amely kielégíti az $A^2 - A - 1 = 0$ egyenletet.
- Ezt átrendezve: $A^2 = A + 1$, amiből teljes indukcióval megmutatható, hogy $k \geq 2$ esetén
 - $A^{k-2} \leq F_k \leq A^{k-1}$
 - Ha $N = F_{k+1}$, akkor $A^{k-1} \leq N$,
 - $k - 1 \leq \log_A N = \frac{\log_2 N}{\log_2 A} \approx 1,44 * \log_2 N$
 - $k \leq 1,44 * \log_2 N + 1$

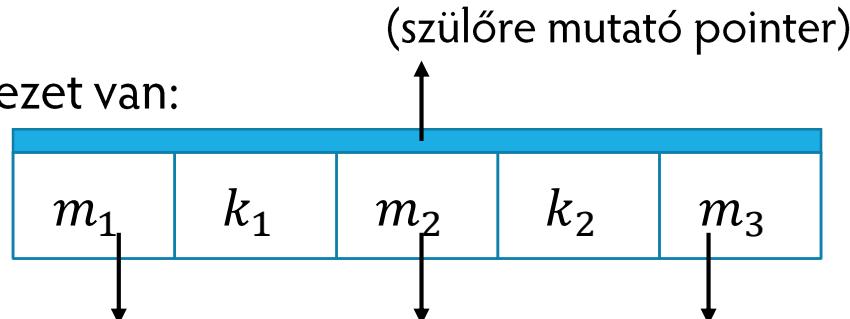


Pázmány Péter Katolikus Egyetem
Információs Technológiai és Bionikai Kar

2-3 fák

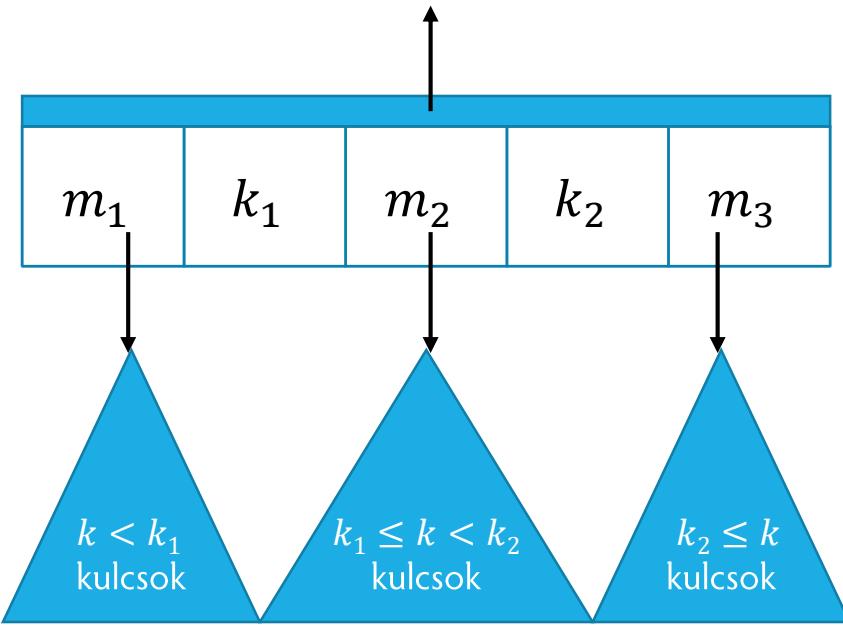
2-3 fák

- Hatékony keresőfa-konstrukció.
- Ez is fa, de a binárisnál bonyolultabb:
egy nem-levél csúcsnak 2 vagy 3 fia lehet.
- A 2-3-fa egy (lefelé) irányított gyökeres fa, melyre:
 - A rekordok a fa leveleiben helyezkednek el, a kulcs értéke szerint balról jobbra növekvő sorrendben. Egy levél egy rekordot tartalmaz.
 - minden belső (azaz nem levél) csúcsból 2 vagy 3 él megy lefelé; ennek megfelelően a belső csúcsok egy, illetve két $k \in U$ kulcsot tartalmaznak.
 - A belső pontokban fizikailag ilyen szerkezet van:



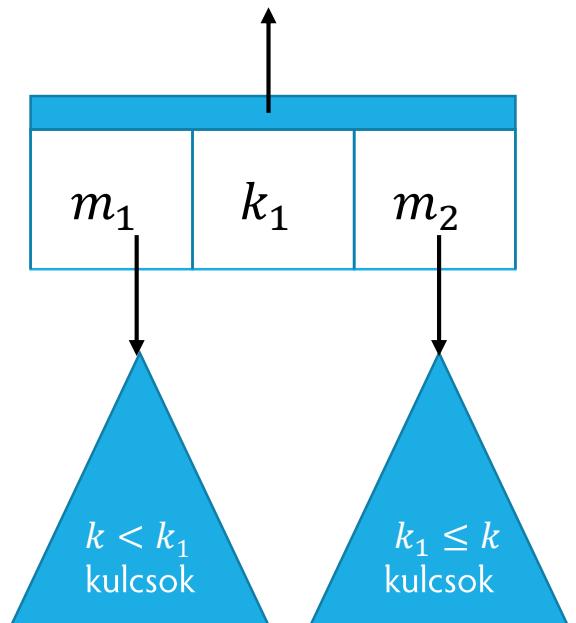
2-3 fák

- Logikailag a belső csúcs kétféle:
 - 3 gyerek
 - Itt $m_1; m_2; m_3$ mutatók a csúcs részfáira, k_1, k_2 pedig U -beli kulcsok, melyekre $k_1 < k_2$.
 - Az m_1 által mutatott részfa minden kulcsa kisebb, mint k_1 .
 - Az m_2 részfájában k_1 a legkisebb kulcs, és minden kulcs kisebb, mint k_2 .
 - Végül m_3 részfájában k_2 a legkisebb kulcs.



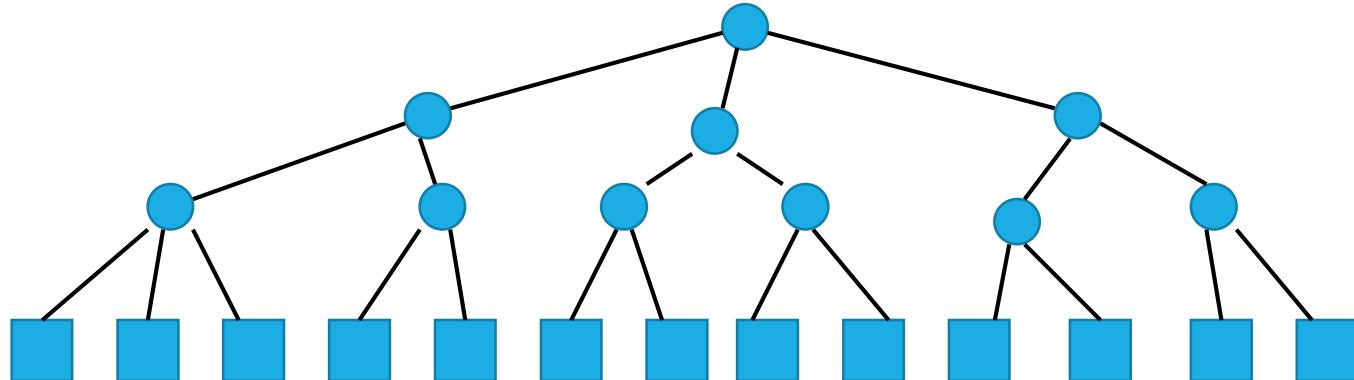
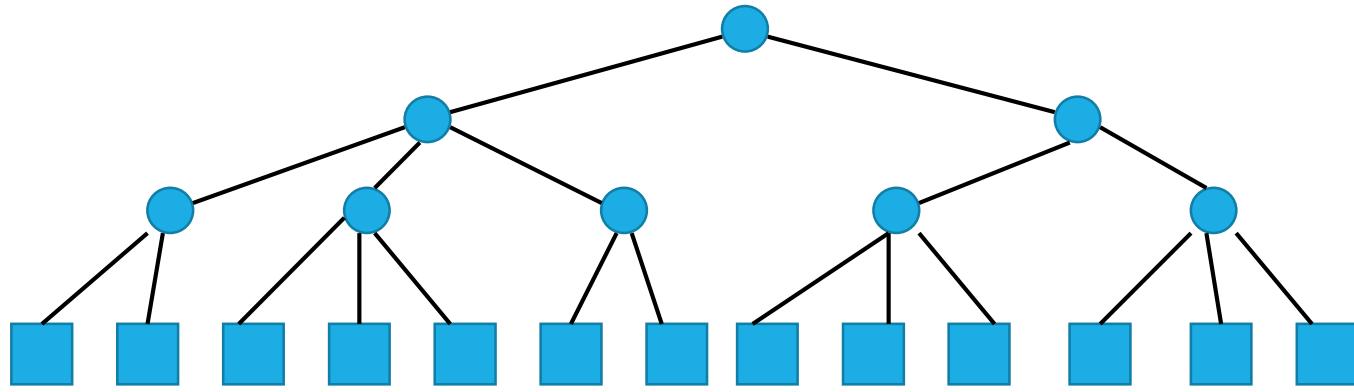
2-3 fák

- Logikailag a belső csúcs kétféle:
 - 2 gyerek
 - Itt m_1, m_2 mutatók a csúcs részfáira, k_1 pedig U -beli kulcs.
 - Az m_1 által mutatott részfa minden kulcsa kisebb, mint k_1 .
 - Az m_2 részfájában k_1 a legkisebb kulcs.
 - A két típus közötti váltást csak logikailag kezeljük.



2-3 fák

- Példa két különböző szerkezetű 2-3 fára, $n = 13$ -ra





2-3 fák

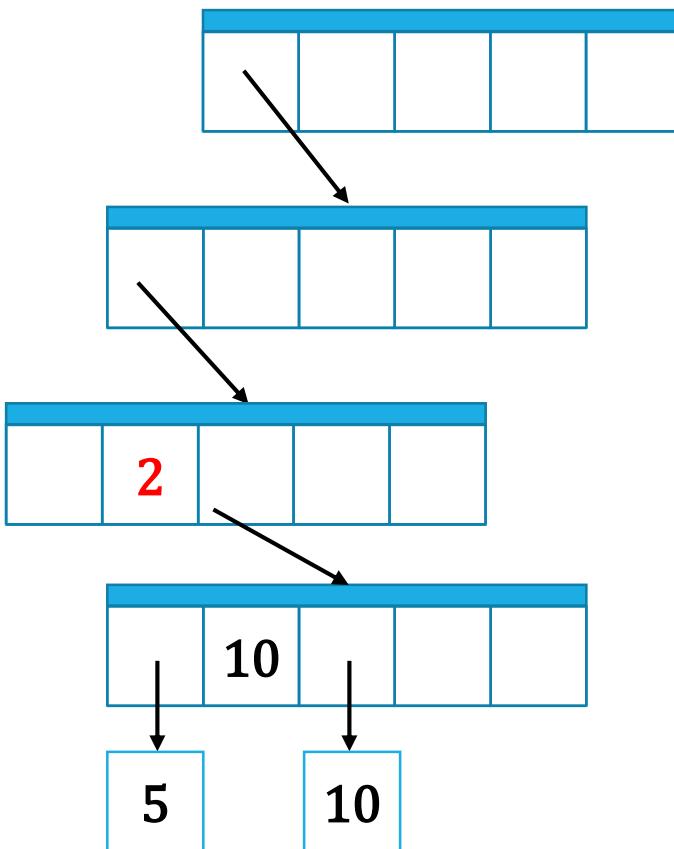
- Megjegyzés:
 - $n = 0$
 - $t = \text{NIL}$ vagy üres gyökér
 - $n = 1$
 - Kivételesen a gyökérnek 1 gyermekéje van
- Összefüggés n és h között:
 - $2^h < n < 3^h \Rightarrow h \leq \log_2 n$
 - Még 2-es elágazási tényező esetén is korlátos a fa magassága!



2-3 fák

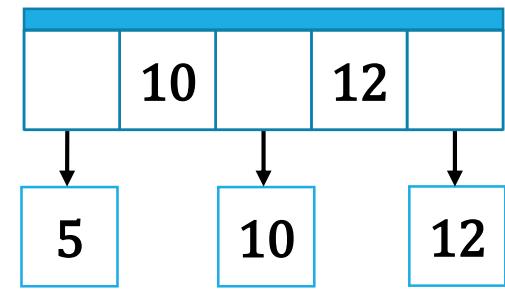
- Műveletek
 - Keresés
 - Összehasonlítások száma
 - $0, 1, \dots, h - 1$ magasságban: 1 vagy 2
 - h magasságban: 1
 - $T(n) < 2h + 1 < 2\log_2 n + 1 = \Theta(\log_2 n)$

2-3 fák – beszúrás

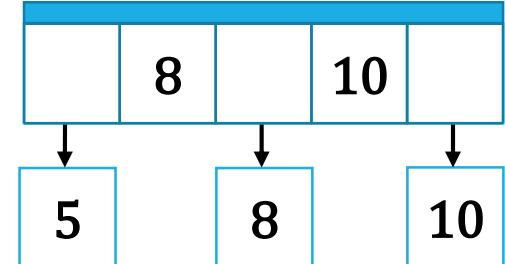


- Kereséssel meghatározzuk a helyét
- I. A legalsó belső pontnak 2 gyereke van (elfér még egy harmadik is)
 - Felfelé haladva korrigálni kell a megfelelő kulcsot 2-re. (ha a nagyszülőben az 5-nél kisebb nem volt, egy leágazásos elem nem szúrható be.)

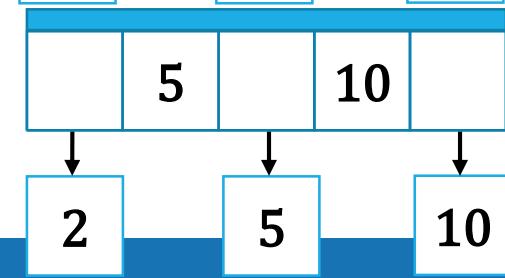
a. $k = 12$



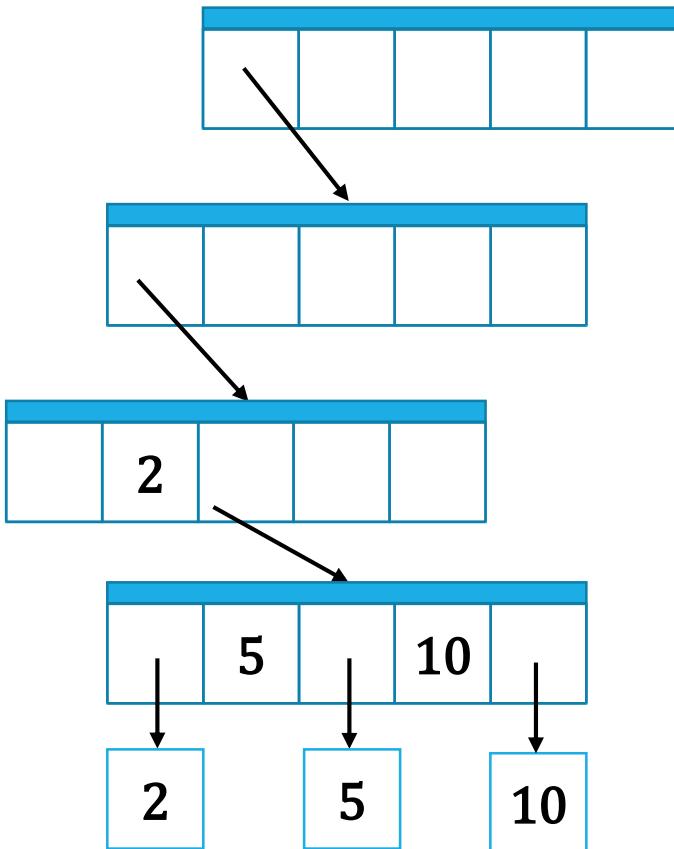
b. $k = 8$



c. $k = 2$



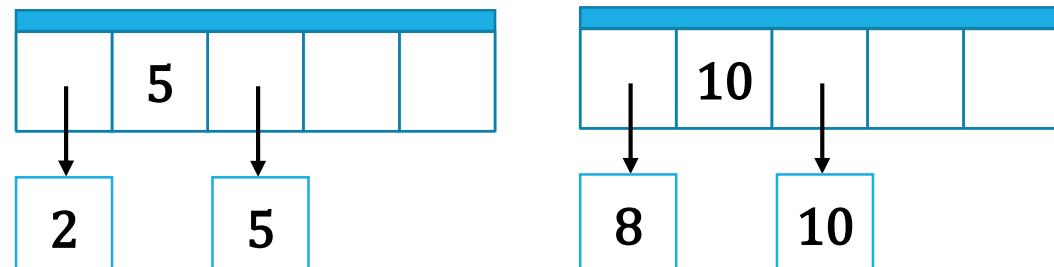
2-3 fák – beszúrás



- Kereséssel meghatározzuk a helyét
- II. A legalsó belső pontnak 3 gyereke van

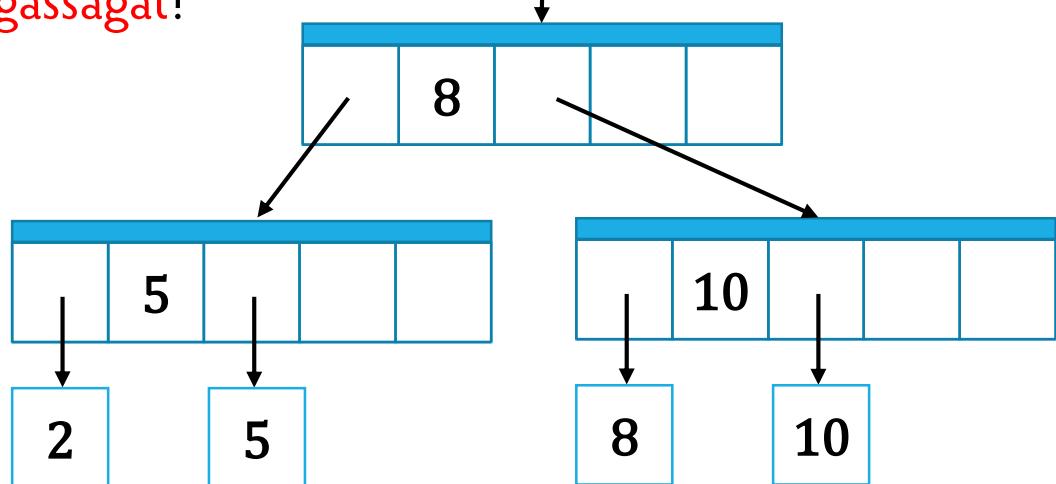
$$k = 8$$

A megoldás a csúcsvágás

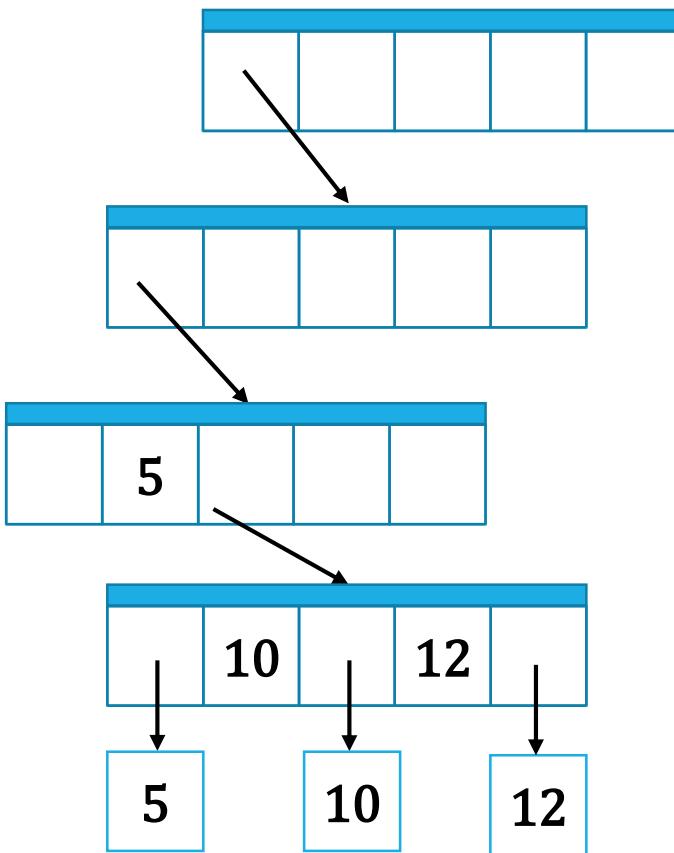


2-3 fák – beszúrás

- II. A legalsó belső pontnak 3 gyereke van. (folyt.)
 - Ha a szülőnek eleve 3 gyereke volt, akkor itt is csúcsvágásra van szükség, és így tovább felfelé. Ha valahol ezen az úton van egy kétgyermekes belső csúcs, akkor ott megáll a beszúrás, mert annak lehet 3 gyereke
 - Ha ezen az úton minden belső pontnak 3 gyereke volt, akkor a csúcsvágás felgyűrűzik a gyökérig. Ekkor a felfelé vágott gyökér fölé egy új gyökeret kell tenni, ami megnöveli a fa magasságát!
 - Az előző példát véve alapul:

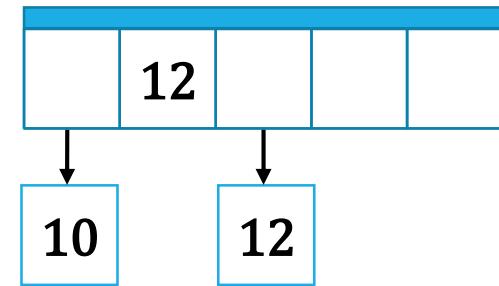


2-3 fák – törlés



- Megkeressük a törlendő kulcsot.
- I. A kulcs szülőjének 3 gyereke van
 - tehát neki 2 testvére van

$k = 5$

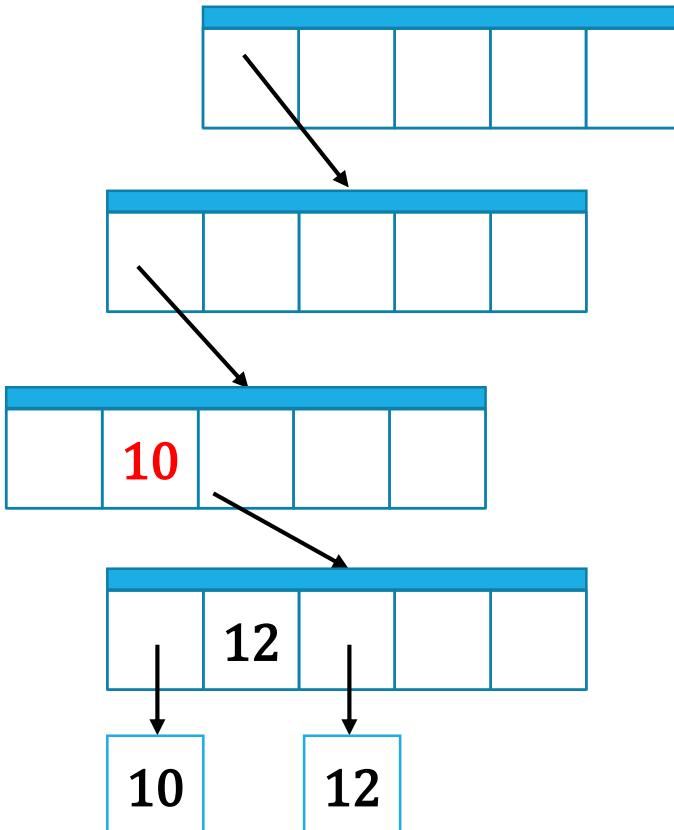


korrekció a szülőben

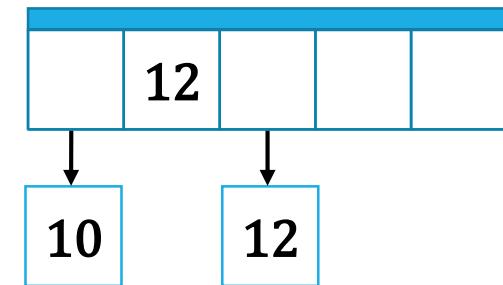
5-ről 10-re

2-3 fák – törlés

- Megkeressük a törlendő kulcsot.
- I. A kulcs szülőjének 3 gyereke van
 - tehát neki 2 testvére van



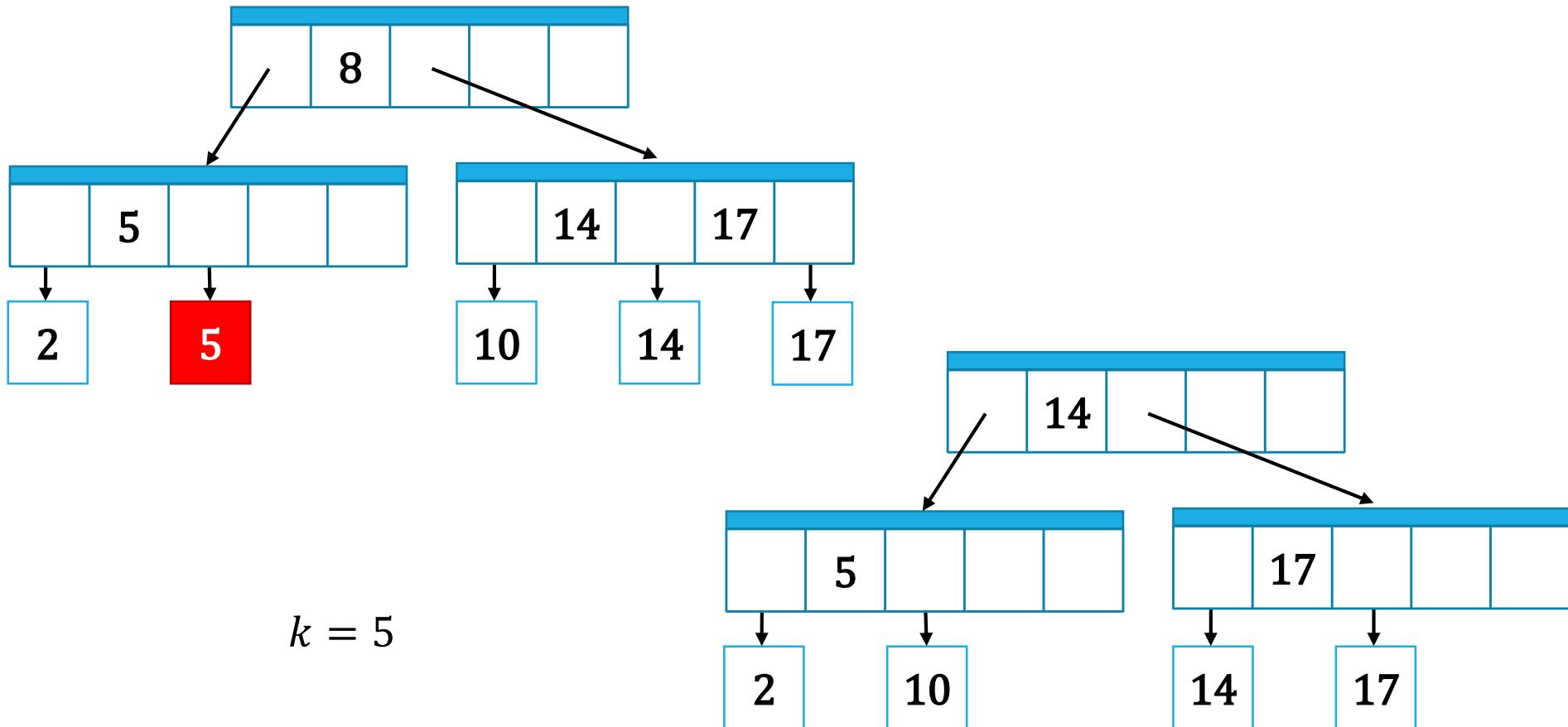
$$k = 5$$



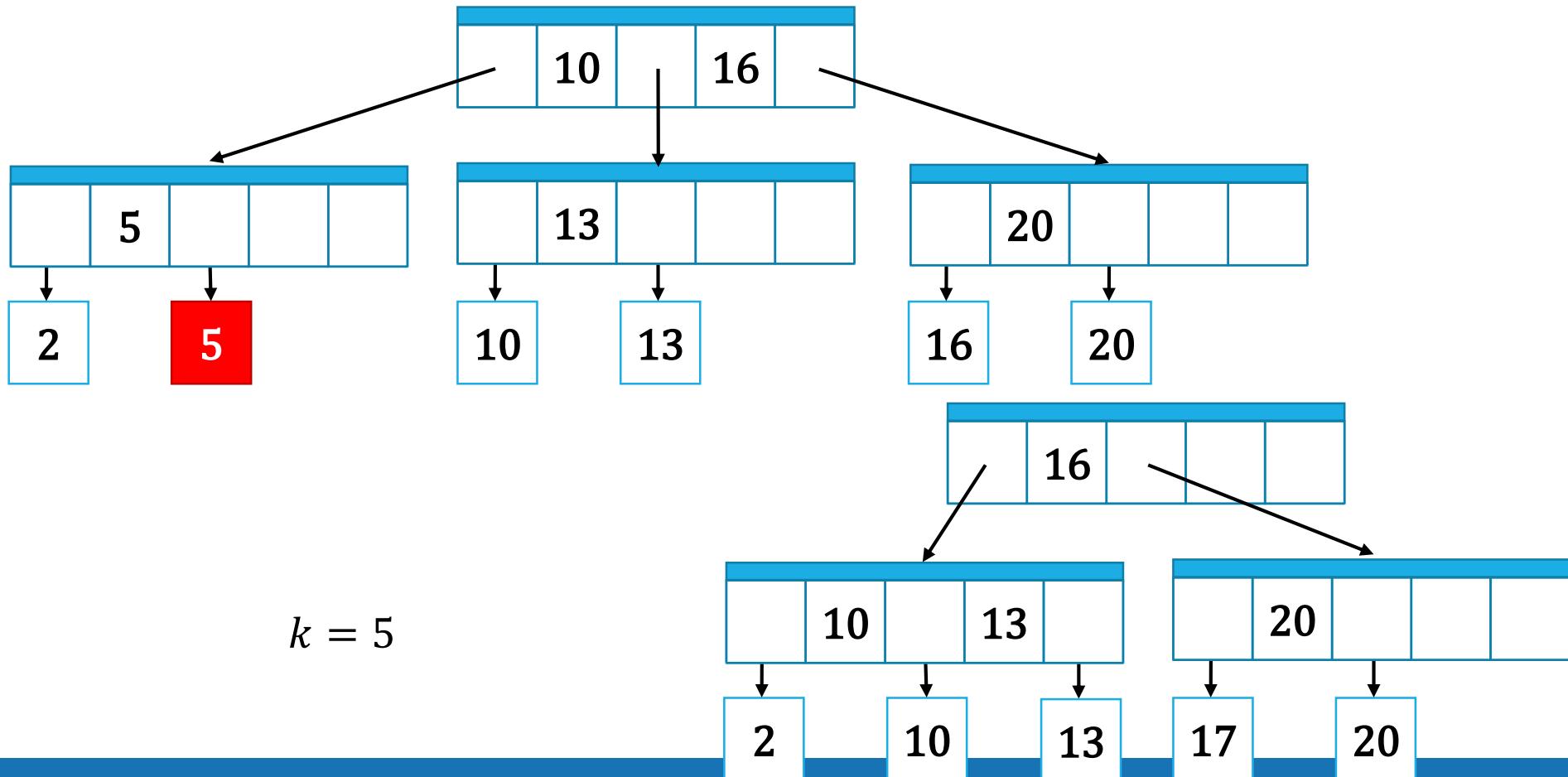
korrekció a szülőben

5-ről 10-re

- II. A törlendő elemnek csak 1 testvére van, (a szülőnek 2 gyereke van)
 - II/1. Ha a szülőnek van 3 gyerekes testvére, akkor az 1 gyereket „átad”

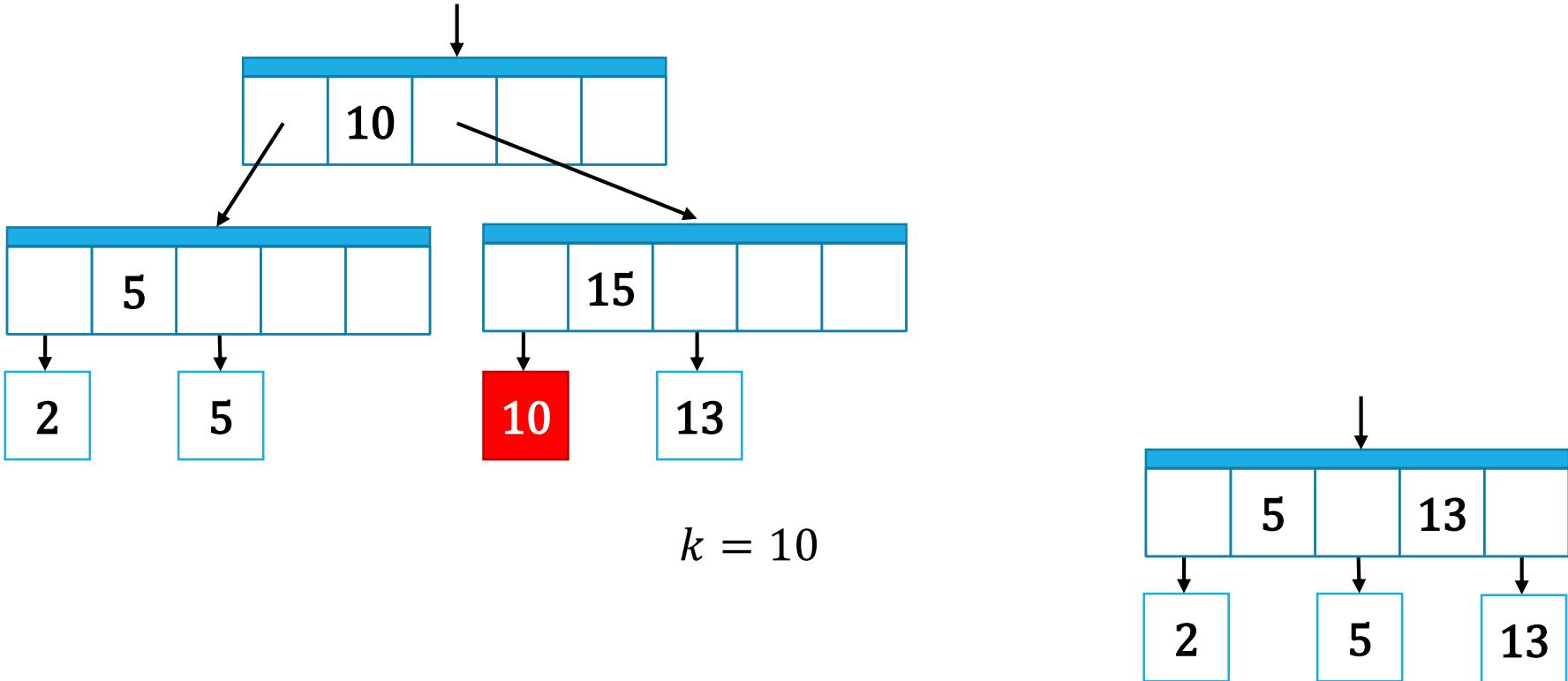


- II. A törlendő elemnek csak 1 testvére van, (a szülőnek 2 gyereke van)
 - II/1. Ha a szülőnek nincs 3 gyerekes testvére, akkor csúcsot vonunk össze



2-3 fák – törlés

- Szükség esetén a csúcsösszevonásokat folytatjuk felfelé.
 - Ez felgyűrűzhet a gyökérig – így a fa magassága csökkenhet





2-3 fák

- Műveletek költsége:
 - $T(n) = \mathcal{O}(h) = \mathcal{O}(\log_2 n)$



Pázmány Péter Katolikus Egyetem
Információs Technológiai és Bionikai Kar

B fák



B-fák

- R. Bayer, E. McCreight, 1972
 - A 2-3-fa általánosításai
 - Nagy méretű adatbázisok, külső táron levő adatok feldolgozására használják.
 - Több szabvány tartalmazza valamilyen változatát
 - B^+ -fa, B^* -fa
 - Probléma, hogy nem az összehasonlítás időigényes, hanem az adatok kiolvasása, de sokszor egy adat kiolvasásához amúgy is kiolvasunk több más adatot, egy lapot
 - A fa csúcsai legyenek lapok, a költség a lapelérések száma



B-fák

- Háttértár műveletek modellezése:

- Legyen x egy objektumra mutató pointer.
- Ha az objektum pillanatnyilag a központi memóriában van, akkor a mezőre a szokásos módon hivatkozhatunk – $x.\text{kulcs}$
- Ha a mágneslemezen van, akkor először a $\text{LEMEZROL_OLVAS}(x)$ kell, ami beolvassa az x által hivatkozott objektumot a központi memóriába, utána lehet csak a mezőre hivatkozni.
- Hasonlóan a $\text{LEMEZRE_IR}(x)$ menti el a megváltozott mezőjű (x által hivatkozott) objektumot a mágneslemezre.
- (Feltesszük, hogy ha x már a memóriában van, akkor $\text{LEMEZROL_OLVAS}(x)$ nem végez lemezolvasást, azaz ekkor egy NOP, „no-operation” műveletnek felel meg)



B-fák

- Egy x objektummal kapcsolatos művelet tipikus mintája:

...

$x \leftarrow$ az objektum mutatója

LEMEZROL_OLVAS(x)

x mezőit olvasó és módosító műveletek

LEMEZRE_IR(x)

- kimarad, ha x egyik mezője sem változott meg

további x mezőit olvasó műveletek

...

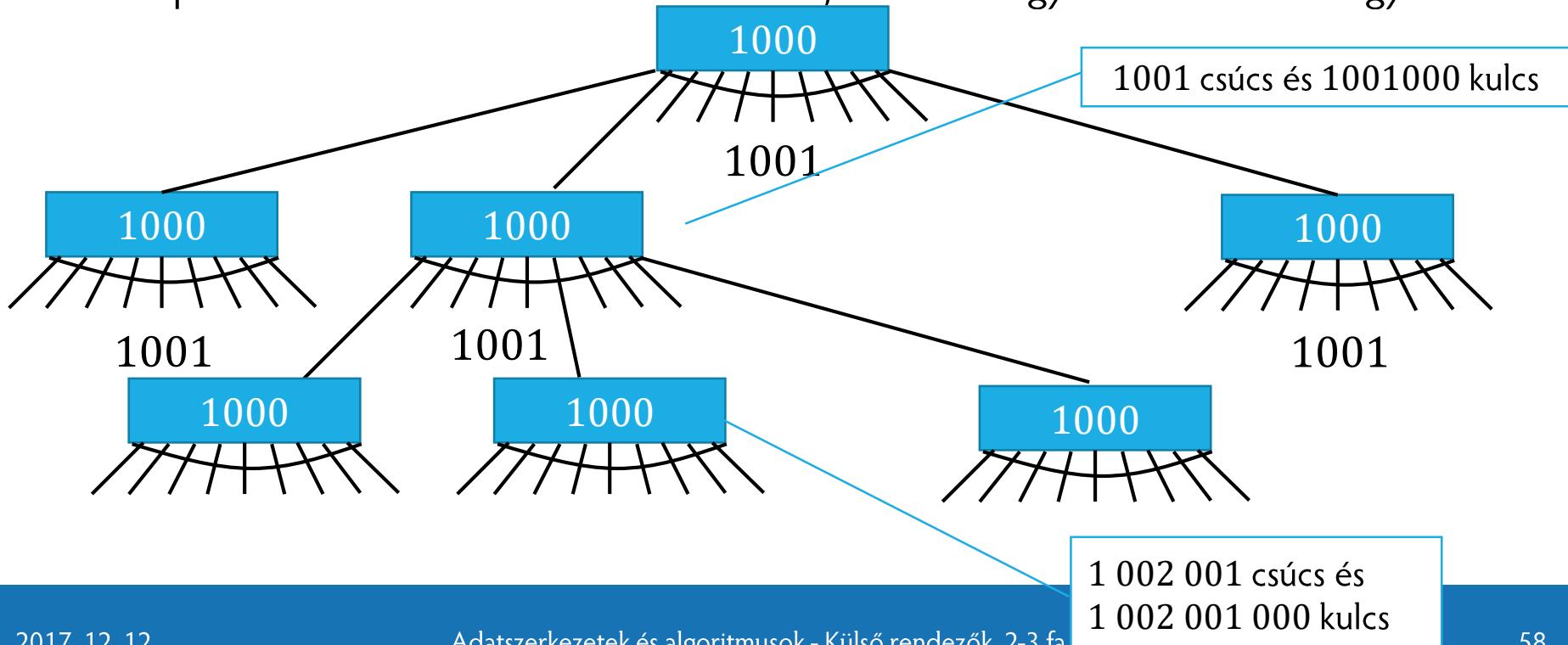


B-fák

- A futási időt a **LEMEZROL_OLVAS(x)** és a **LEMEZRE_IR(x)** műveletek száma határozza meg.
 - Tehát egy művelet annyit (olyan keveset) olvasson, illetve írjon, amennyit csak lehet!
 - Azaz a B-fa egy csúcsának a nagysága a mágneslemez egy lapjának a méretének felel meg
 - Az elágazási tényező 50 és 2000 között
 - Így a fa magassága jelentősen csökken.

Példa

- 2-magasságú B-fa, elágazási faktora 1001, több, mint 1 milliárd kulcs!
 - A gyökeret állandóan a központi memóriában tartva bármelyik kulcs eléréséhez maximum 2 lemezművelet kell!
 - A példában 1 csúcs 1000 kulcsot tartalmaz, tehát 1001 gyermeket van mindeneknek





B-fák – Definíció

- Feltételezés:
 - A kulcshoz tartozó minden „kísérő információ” ugyanabban a csúcsban, mint a kulcs (minden kulcshoz egy pointer arra a lapra, ahol a többi inf.)
 - Ekkor feltesszük, hogy ha a kulcsot mozgatjuk, ezt a pointert visszük magunkkal
 - minden „kísérő információ” a levelekben
 - Ez az úgymevezett B+ fa – itt csak a kulcsok és a gyerekekre mutató pointerek vannak a közbülső csúcsokban – és a levelek is össze vannak linkelve



B-fák – Definíció

- B-fa definíciója:
 1. minden x csúcsnak a következő mezői vannak:
 - $n[x]$ – az x csúcsban tárolt kulcsok darabszáma
 - az $n[x]$ darab kulcs, a kulcsokat monoton növekvő sorrendben tároljuk:
 - $x.kulcs_1 < x.kulcs_2 < \dots < x.kulcs_{n[x]}$
 - $x.level$ – logikai változó, IGAZ, ha x levél, HAMIS, ha x egy belső csúcs
 2. Ha x egy belső csúcs (nem levél), akkor tartalmazza a $x.c_1, x.c_2, \dots, x.c_{n[x]+1}$ mutatókat az x gyerekeire
 - A levél csúcsoknak nincsenek gyerekeik, a levelek $x.c_i$ mutatói definiálatlanok



B-fák – Definíció

3. Az $x.kulcs_i$ értékek meghatározzák a kulcsértékeknek azokat a tartományait, amelyekbe a részfák kulcsai esnek
$$k_1 \leq x.kulcs_1 < k_2 \leq x.kulcs_2 < \dots \leq x.kulcs_{n[x]} < k_{n[x]+1}$$
4. **Minden levélnek azonos a mélysége**, ez az érték a fa h magassága
5. A csúcsokban tárolható kulcsok darabszámára adott egy **alsó** és egy **felső** korlát. Ezeket a korlátokat egy $t \geq 2$ egész számmal lehet kifejezni, ezt a számot nevezzük a B-fa minimális fokszámának

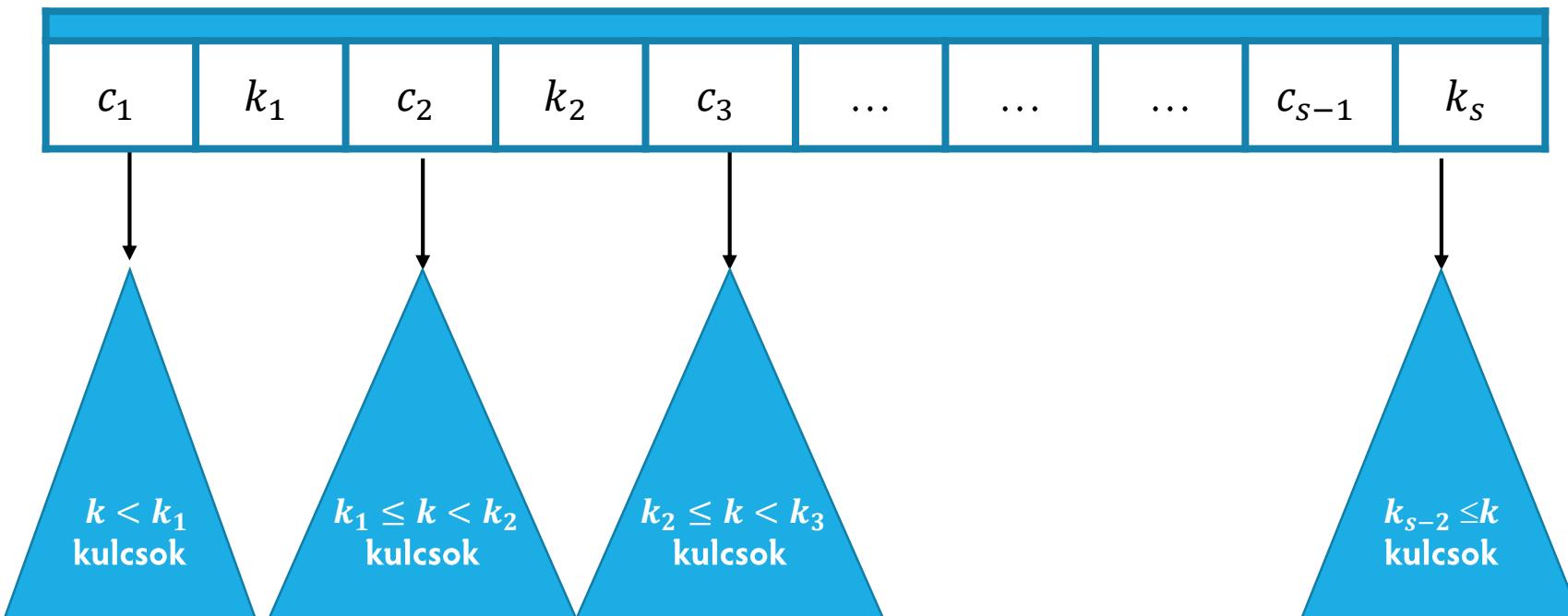


B-fák – Definíció

- minden nem gyökér csúcsnak legalább $t - 1$ kulcsa van, minden belső csúcsnak így legalább t gyereke van. Ha a fa nem üres, akkor a gyökérnek legalább egy kulcsa kell legyen
- minden csúcsnak legfeljebb $2t - 1$ kulcsa lehet, tehát egy belső csúcsnak legfeljebb $2t$ gyereke lehet. Egy csúcs **telített**, ha pontosan $2t - 1$ kulcsa van

B-fák

- A belső csúcsok hasonlítanak a 2-3-fák belső csúcsaira.
- Egy belső csúcs logikailag így néz ki





B-fák

- A B-fa magassága

- **Tétel:** Ha $n \geq 1$, akkor minden olyan T n -kulcsos B-fára, amelynek h a magassága és minimális fokszáma $t \geq 2$, teljesül, hogy

$$h \leq \log_t \frac{n+1}{2}$$



B-fák

- A B-fa magassága

- **Bizonyítás:** Ha egy B-fa magassága h , akkor csúcsainak száma akkor minimális, ha a gyökércsúcsnak 1 kulcsa, minden más csúcsnak $t - 1$ kulcsa van. Ekkor van 2 darab 1-mélységű, $2t$ darab 2-mélységű, $2t^2$ darab 3-mélységű, ..., $2t^{h-1}$ darab h -mélységű csúcs.

- Így a kulcsok n darabszámára teljesül:

$$n \geq 1 + (t - 1) \sum_{i=1}^h 2t^{i-1} = 1 + 2(t - 1) \left(\frac{t^h - 1}{t - 1} \right) = 2t^h - 1$$



A B-fa műveletei – Keresés

- Keresés, beszúrás és törlés: a keresőfák, illetve a 2-3 fák alapján könnyen elképzelhető.
- Feltételek:
 1. A B-fa gyökere mindenkor a központi memóriában van, így a gyökércsúcsra **LEMEZROL-OLVAS** művelet nem kell, de **LEMEZRE-IR** kell akkor, ha a gyökércsúcs megváltozott
 2. minden olyan csúcs, amely paraméterként szerepel, már a központi memóriában van, már végrehajtottunk rá egy **LEMEZROL-OLVAS** műveletet



A B-fa műveletei – Keresés

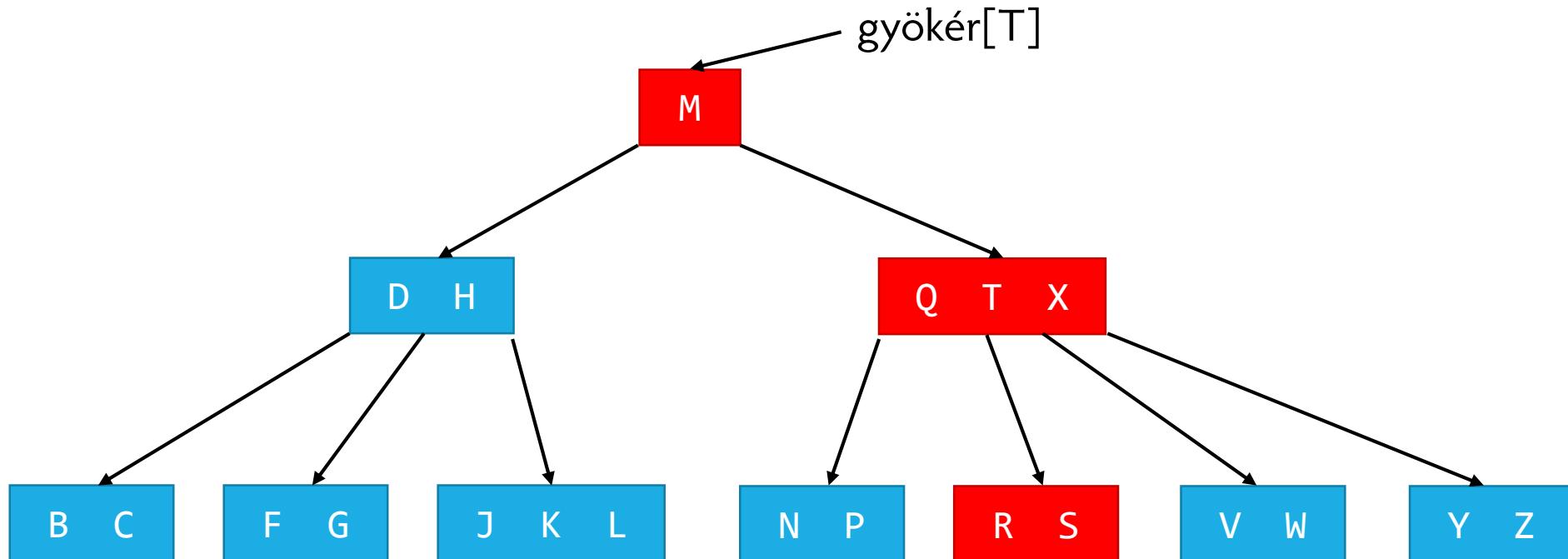
- Keresés: itt minden belső csúcsban $n[x]+1$ lehetőséget kell megvizsgálni.
 - x a részfa gyökércsúcsára mutató pointer, k a kulcs, amit ebben a részfában keresünk:

B-FABAN-KERES(x, k)

```
i←1
while i≤n[x] és k>x.kulcsi do
    i←i+1
    if i≤n[x] és k=x.kulcsi
        then return (x,i)
    if x.levél
        then return NIL
    else LEMEZROL-OLVAS(x.ci)
        return B-FABAN-KERES(x.ci,k)
```

A B-fa műveletei – Keresés

- Ha az R betűt keressük, a piros színnel jelzett csúcsokon haladunk végig





A B-fa műveletei – Keresés

- A B-FÁBAN-KERES lemezműveleteinek száma
 $\Theta(h) = \Theta(\log_t n)$
- Mivel $n[x] < 2t$, így a while ciklus ideje minden csúcsra
 $\Theta(t)$
- A központi egység összes műveleti ideje
 $\Theta(t * h) = \Theta(t * \log_t n)$



A B-fa műveletei – Létrehozás

- **B-FAT-LETREHOZ** – egy üres gyökércsúcsot ad.
 - Használja a PONTOT-ELHELYEZ eljárást, ez $\mathcal{O}(1)$ idő alatt lefoglalja az új csúcsnak a lemez egy lapját.
 - Feltételezzük, hogy nincs szüksége a LEMEZROL-OLVAS eljárás meghívására

B-FAT-LETREHOZ(T)

$x \leftarrow \text{PONTOT-ELHELYEZ}()$

$x.\text{levél} \leftarrow \text{IGAZ}$

$n[x] \leftarrow 0$

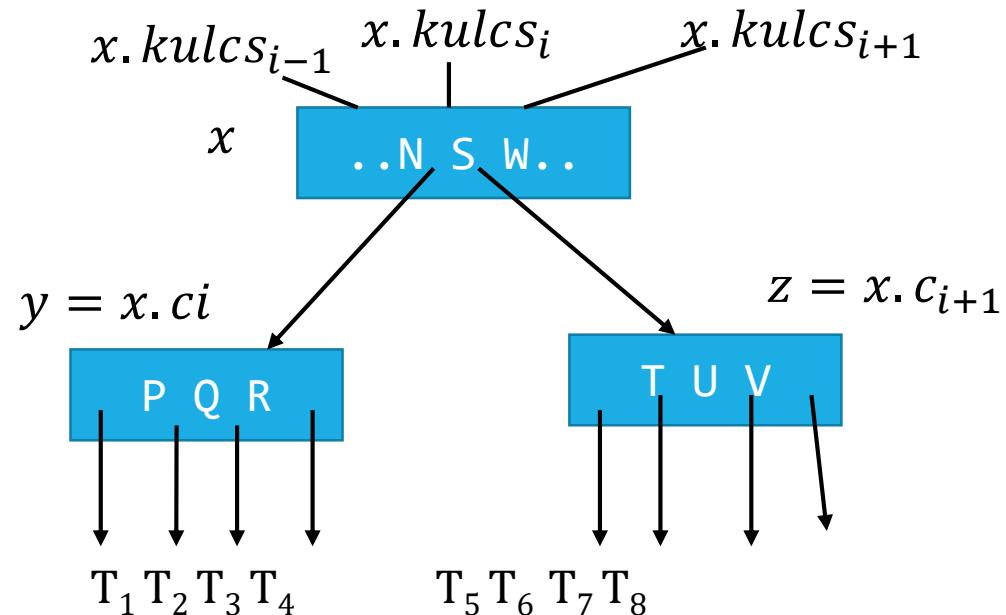
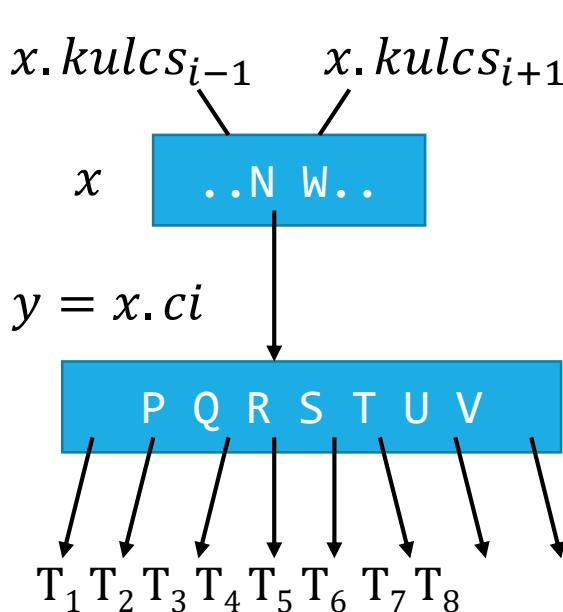
$\text{LEMEZRE-IR}(x)$

$\text{gyökér}[T] \leftarrow x$

- Ehhez $\mathcal{O}(1)$ lemezművelet és $\mathcal{O}(1)$ központi egység idő kell.

A B-fa műveletei – Csúcs szétvágása

- A telített, $2t - 1$ darab kulcsot tartalmazó y csúcsot szétvágjuk középső kulcsa, $y.kulcs_t$ körül két $t - 1$ kulcsú csúcsra
 - A középső csúcs átmegy az y szülőjébe – ez még nem volt telített az y szétvágása előtt
 - Ha y -nak nincs szülője, akkor a fa magassága eggyel nő.





A B-fa műveletei – Csúcs szétvágása

- Tegyük fel, hogy x egy nem telített belső csúcs, $y = x.c_i$, és y az x -nek egy telített gyereke:

B-FA-VAGAS-GYEREK(x, i, y)

$z \leftarrow \text{PONTOT-ELHELYEZ}()$

- $\mathcal{O}(1)$ idő alatt lefoglalja az új csúcsnak a lemez egy lapját

$z.\text{levél} \leftarrow y.\text{levél}$

$n[z] \leftarrow t-1$

for $j \leftarrow 1$ to $t-1$ **do**

- Átmásoljuk bele az y „végét”

$z.\text{kulcs}_j \leftarrow y.\text{kulcs}_{j+t}$

if not $y.\text{levél}$

then **for** $j \leftarrow 1$ to t **do**

$z.c_j \leftarrow y.c_{j+t}$

$n[y] \leftarrow t-1$

- most már y mérete is $t - 1$



A B-fa műveletei – Csúcs szétvágása

```
for j←n[x]+1 downto i+1 do
    x.cj+1←x.cj
```

$x.c_{i+1} \leftarrow z$

```
for j←n[x] downto i do
    x.kulcsj+1←x.kulcsj
```

$x.kulcs_{i+1} \leftarrow y.kulcs_t$

$n[x] \leftarrow n[x] + 1$

LEMEZRE-IR(y)

LEMEZRE-IR(z)

LEMEZRE-IR(x)

a csúcsokra mutatókat arrébb tesszük x -ben
z középre
a kulcsokat arrébb tesszük

az y középső kulcsa a helyére
 x elemszáma nőtt



A B-fa műveletei – Beszúrás

- Beszúrás: Egy k kulcs beszúrása egy h magasságú T B-fába egy egymenletes, a fában lefelé haladó algoritmussal oldható meg, a végrehajtáshoz $\mathcal{O}(h)$ lemezhozzáférés kell
 - A szükséges központi egység idő $\mathcal{O}(t * h) = \mathcal{O}(t \log_t n)$

B-FABA-BESZUR(T, k)

$r \leftarrow \text{gyökér}[T]$
 $\text{if } n[r] = 2t - 1$

- ha telített a gyökércsúcs, vág

$\text{then } s \leftarrow \text{PONTOT-ELHELYEZ}()$

$\text{gyökér}[T] \leftarrow s$
 $s.\text{levél} \leftarrow \text{HAMIS}$
 $n[s] \leftarrow 0$
 $s.c1 \leftarrow r$

B-FA-VÁGÁS-GYEREK(s, 1, r)

NEM-TELITETT-B-FABA-BESZUR(s, k)

else NEM-TELITETT-B-FABA-BESZUR(r, k)



A B-fa műveletei – Beszúrás

NEM-TELITETT-B-FABA-BESZUR(x,k)

$i \leftarrow n[x]$

if x .levél

then

while $i \geq 1$ és $k < x$.kulcs_i do

x .kulcs_{i+1} $\leftarrow x$.kulcs_i

$i \leftarrow i - 1$

x .kulcs_{i+1} $\leftarrow k$

$n[x] \leftarrow n[x] + 1$

LEMEZRE-IR(x)

else

while $i \geq 1$ és $k < x$.kulcs_i do

$i \leftarrow i - 1$

$i \leftarrow i + 1$

LEMEZROL-OLVAS($x.c_i$)

...

hátulról kezdjük

itt a k helye
x mérete nő

ha nem levél,
megkeressük a helyét



A B-fa műveletei – Beszúrás

NEM-TELITETT-B-FABA-BESZUR(x, k)

...

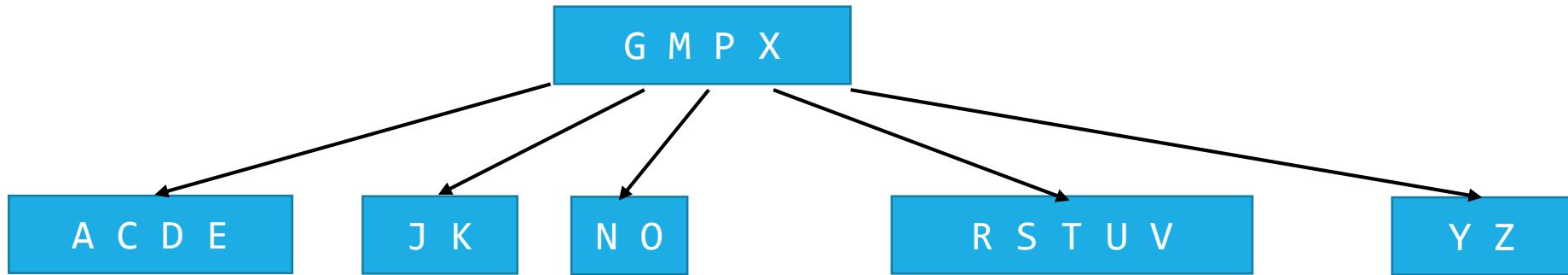
```
if n[x.ci] = 2t - 1
  then B-FA-VAGAS-GYEREK(x, i, x.ci)
      if k > x.kulcsi
          then i ← i + 1
```

Telített?

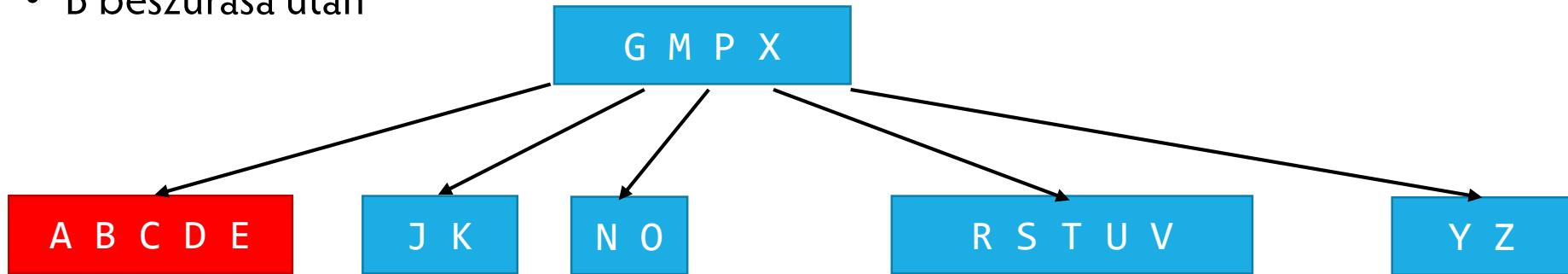
NEM-TELITETT-B-FABA-BESZUR($x.c_i, k$)

A B-fa műveletei – Beszúrás

- Tegyük fel, hogy $t = 3$, azaz maximum 5 kulcs lehet egy csúcsban

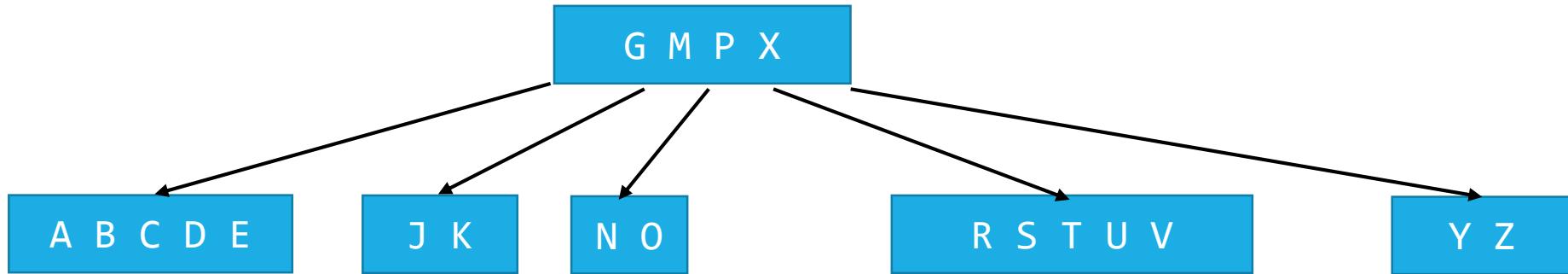


- B beszúrása után

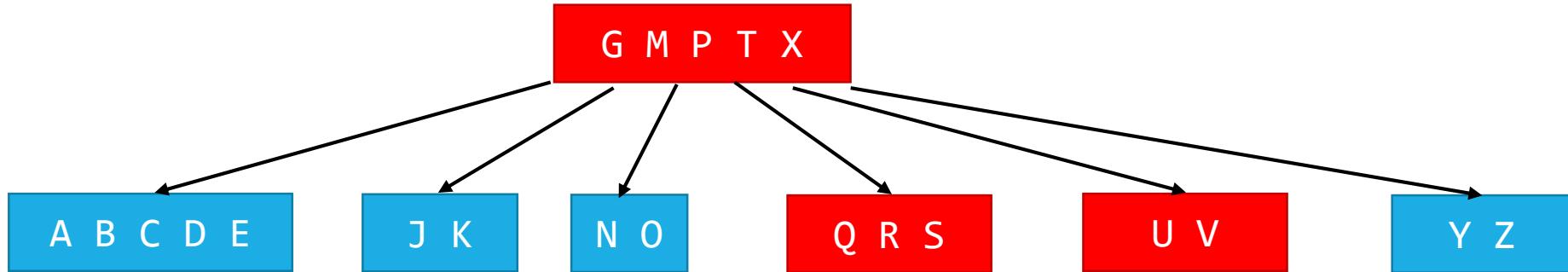


A B-fa műveletei – Beszúrás

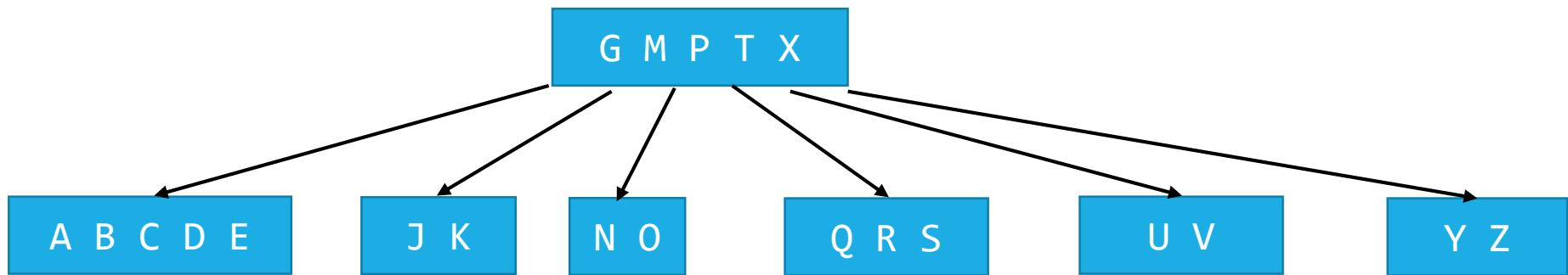
- Tegyük fel, hogy $t = 3$, azaz maximum 5 kulcs lehet egy csúcsban



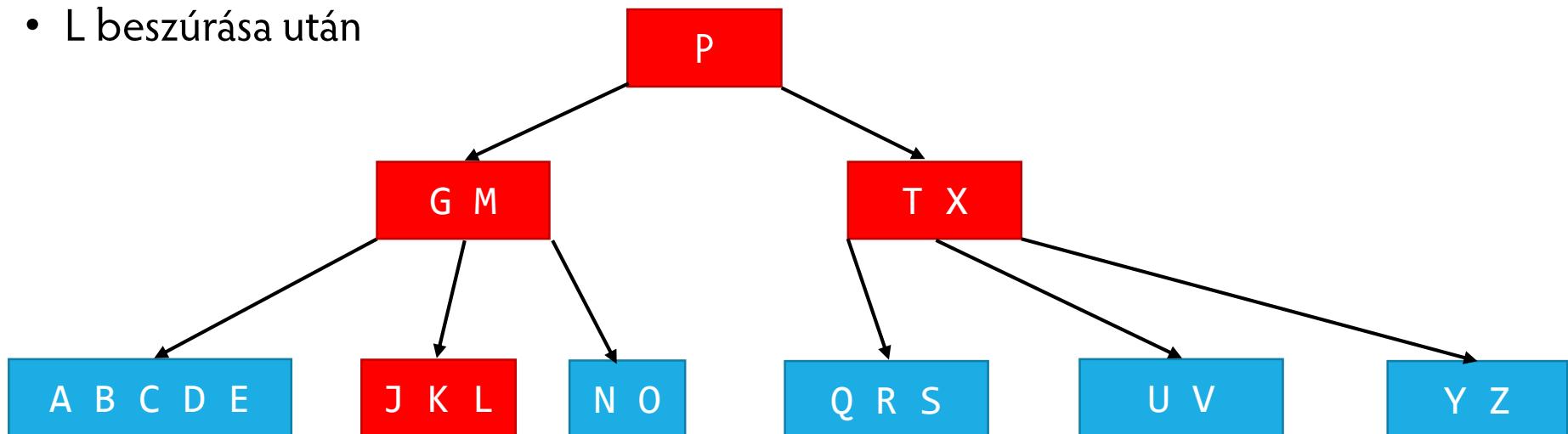
- Q beszúrása után (Az RSTUV csúcs telített)



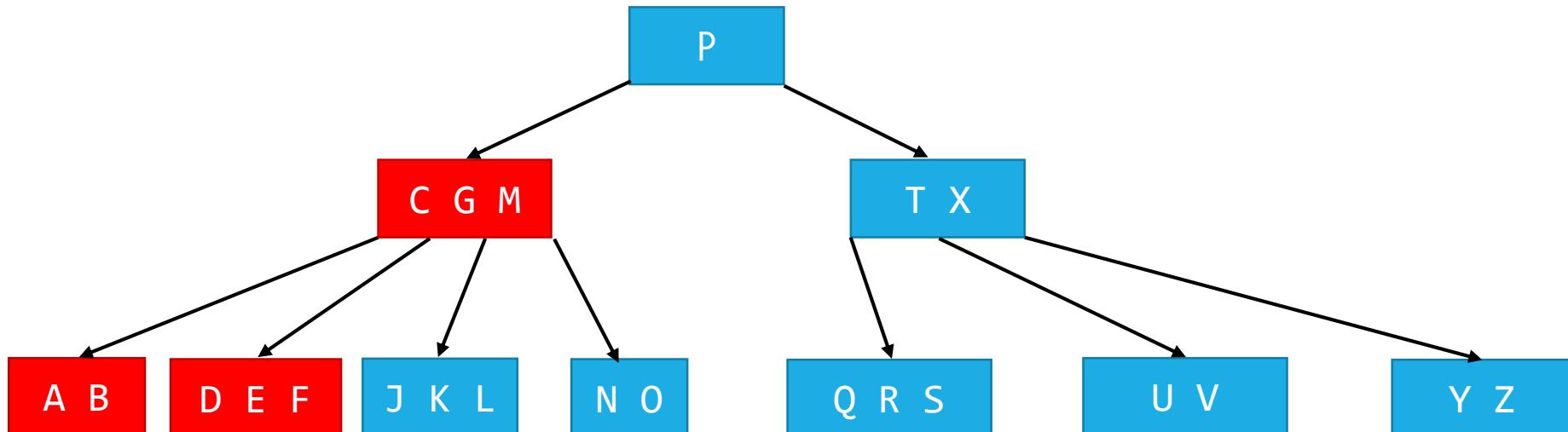
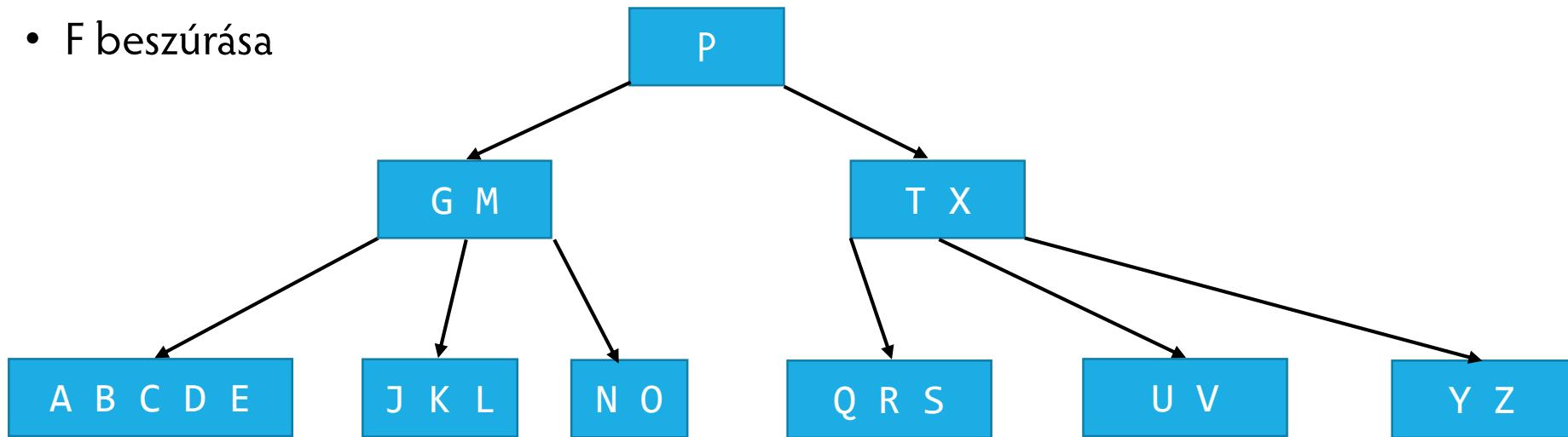
- Tegyük fel, hogy $t = 3$, azaz maximum 5 kulcs lehet egy csúcsban



- L beszúrása után



- F beszúrása

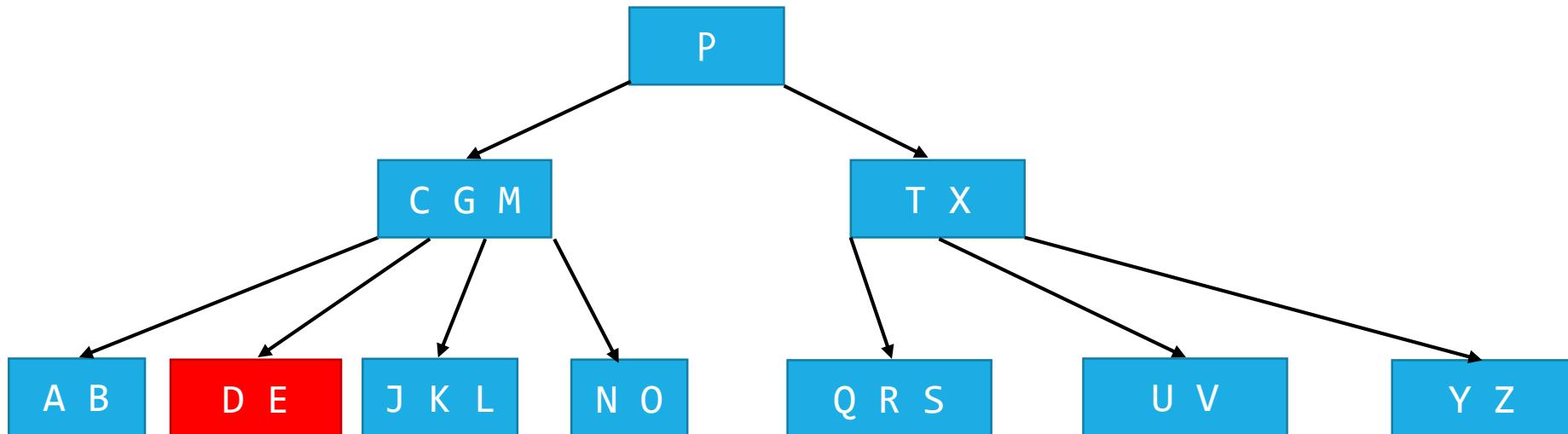
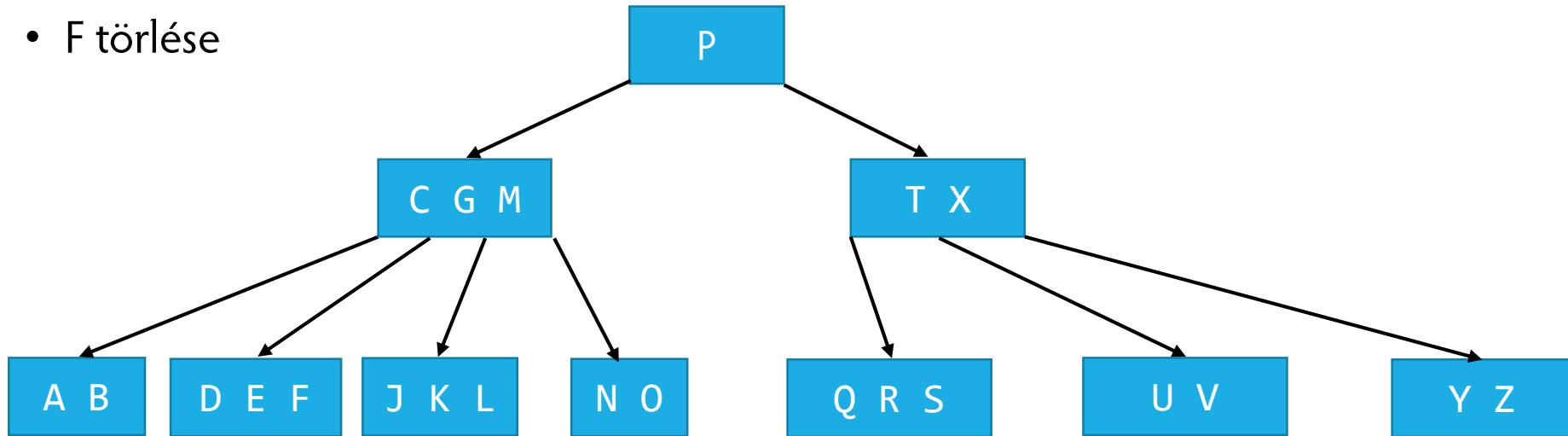




A B-fa műveletei – Törlés

- Törlés – kulcsot nemcsak levélből, hanem tetszőleges csúcsból lehet törlni. Ügyelni kell arra, hogy a csúcs ne váljon túl kicsivé (kivéve a gyökérben)
- Lehetőségek:
 1. A k kulcs az x csúcsban van, x egy levél, akkor a k kulcsot törljük az x -ből

- F törlése

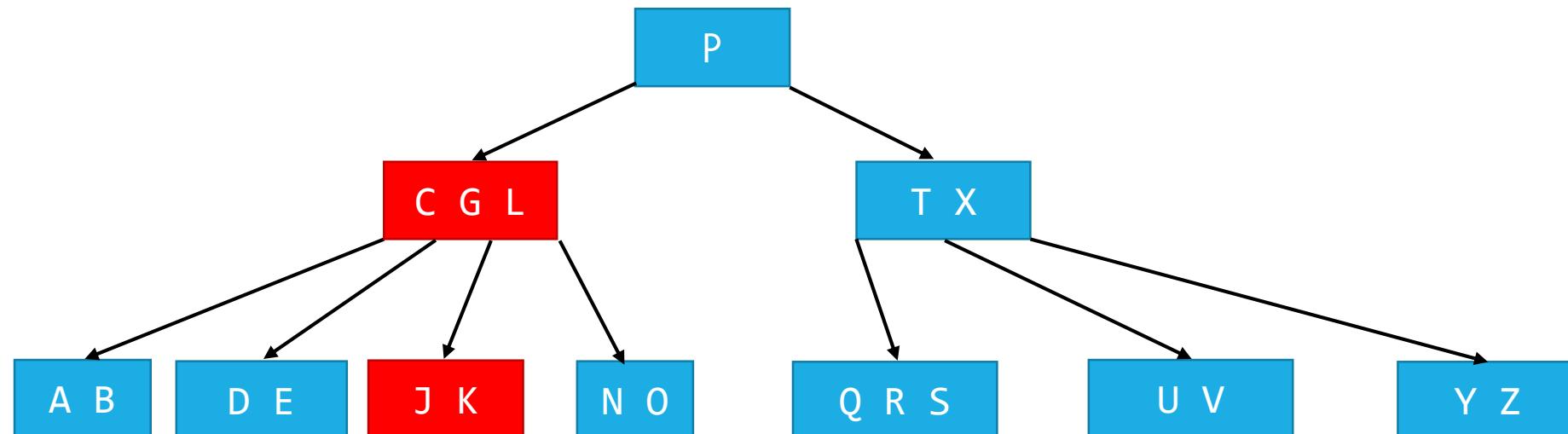
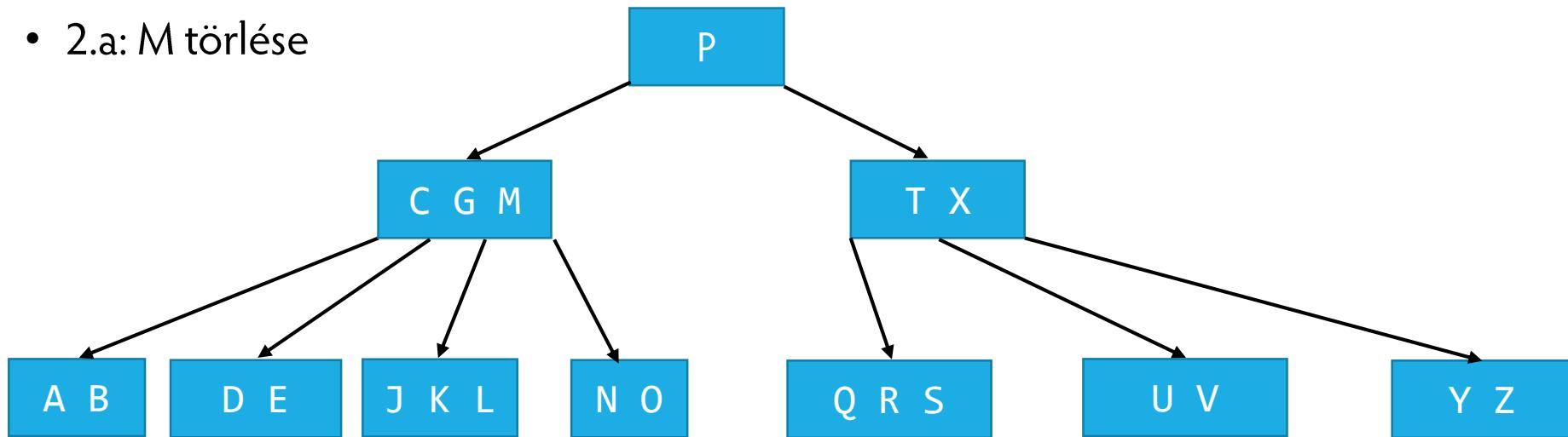




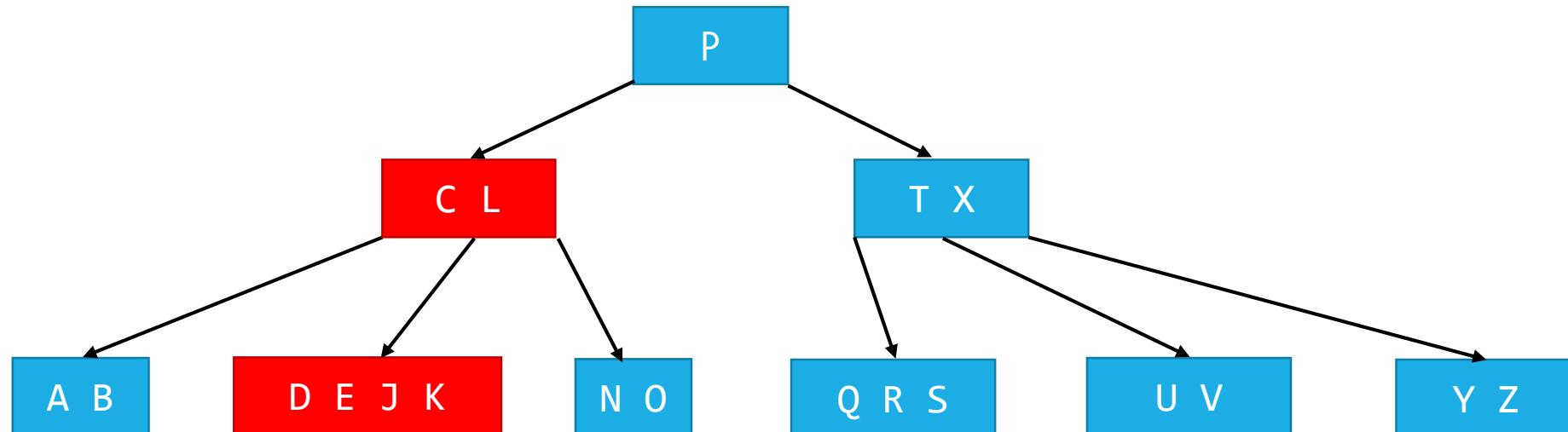
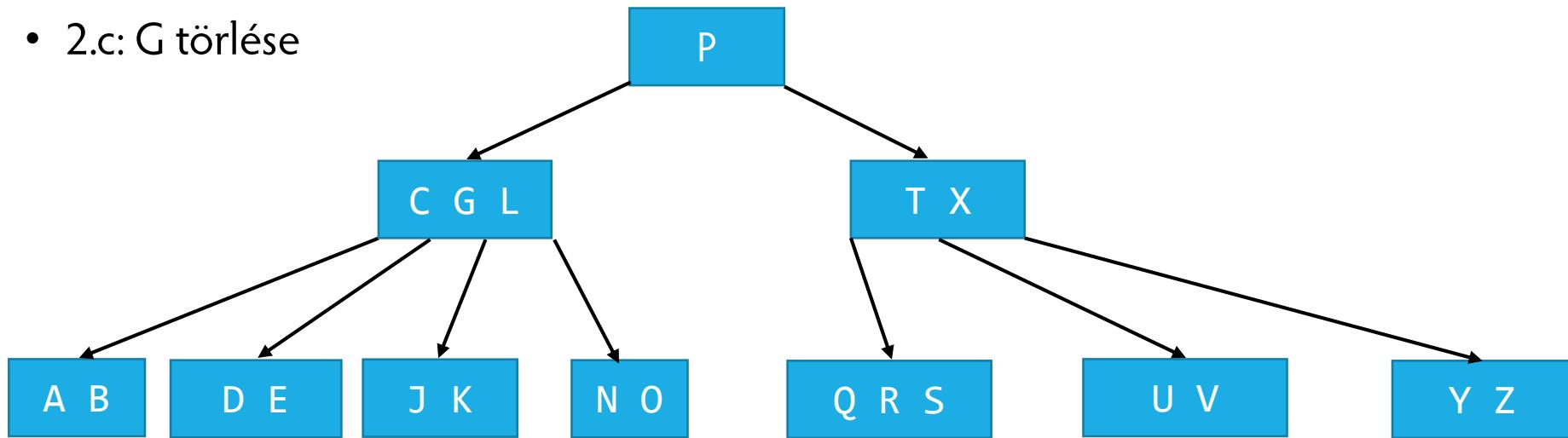
A B-fa műveletei – Törlés

- Lehetőségek:
 2. A k kulcs az x csúcsban van, x a fa egy belső csúcsa, akkor:
 - a. Ha x -ben a k -t megelőző gyereknek (y) legalább t kulcsa van, akkor megkeressük az y részfában a k -t közvetlenül megelőző k' kulcsot. Rekurzívan töröljük k' -t és helyettesítsük k -t k' -vel az x -ben.
 - b. Szimmetrikusan, ha a z gyerek következik az x -beli k után, és z -nek legalább t kulcsa van, akkor keressük meg a z gyökércsúcsú részfában a k -t közvetlenül követő k' kulcsot. Rekurzívan töröljük k' -t és helyettesítsük k -t k' -vel az x -ben.
 - c. Ha mind y -nak, mind z -nek csak $t - 1$ kulcsa van, akkor egyesítsük k -t és z kulcsait y -ba úgy, hogy x -ből töröljük a k -t és a z -re mutató pointert. Ekkor y -nak $2t - 1$ kulcsa lesz. Ezután szabadítsuk fel z -t és rekurzívan töröljük k -t az y -ból.

- 2.a: M törlése



- 2.c: G törlése

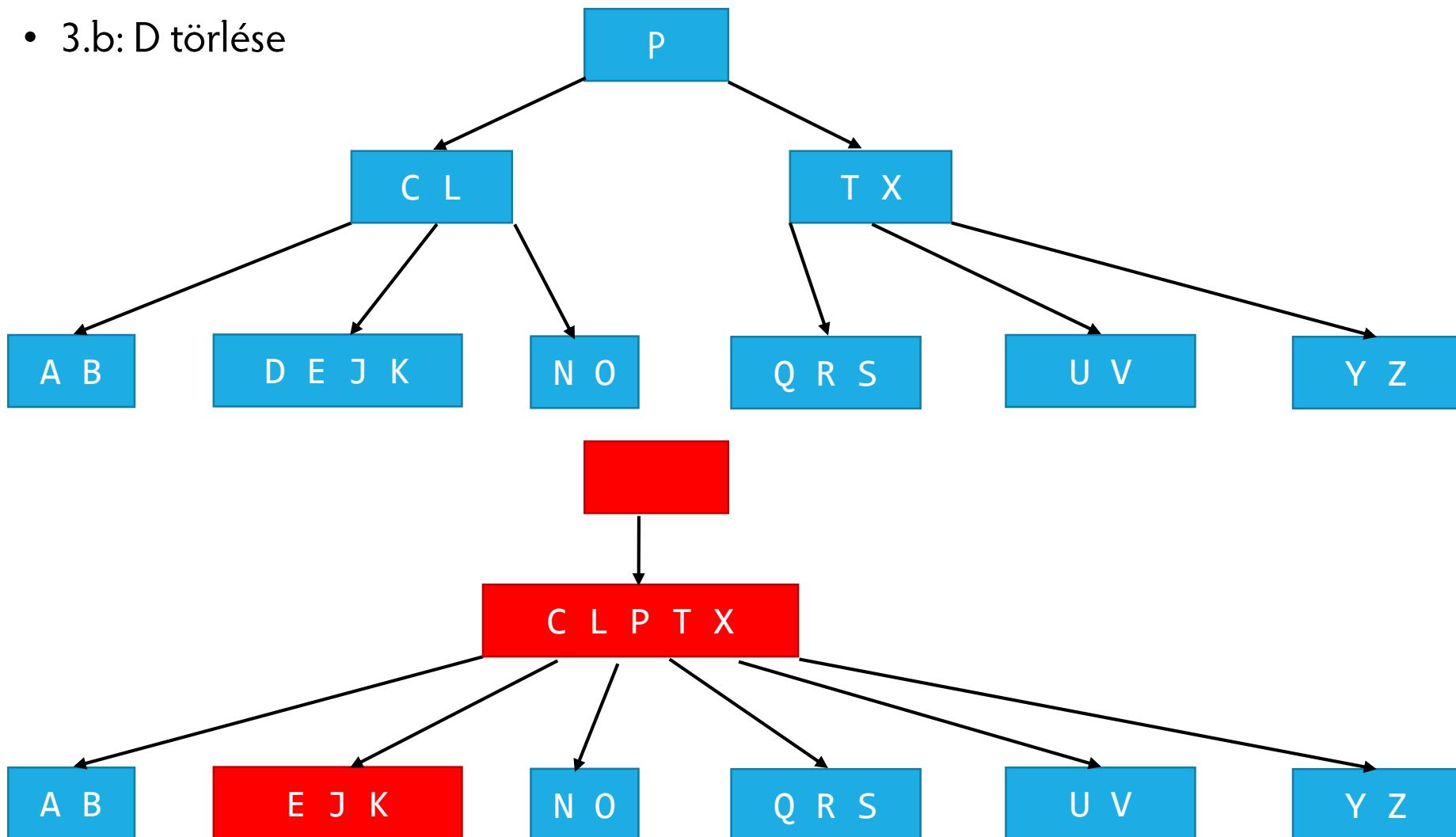




A B-fa műveletei – Törlés

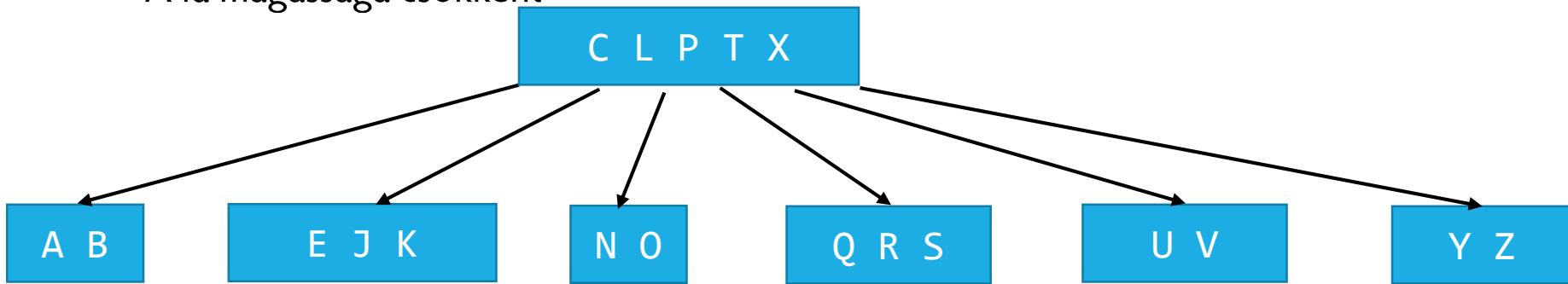
- Lehetőségek
 - 3. Ha a k kulcs nincs benne az x belső csúcsban, akkor határozzuk meg annak a részfának az $x.c_i$ gyökércsúcsát, amelyikben lehet a k , ha egyáltalán szerepel. Ha $x.c_i$ -nek csak $t - 1$ csúcsa van, akkor a 3a vagy 3b szerint járunk el, mivel biztosítani kell, hogy annak a csúcsnak, amelyikre lelépünk, legalább t csúcsa legyen. Ezután rekurzióval megyünk tovább
 - a. Ha $x.c_i$ -nek csak $t - 1$ csúcsa van, de van egy közvetlen testvére, amelyiknek legalább t csúcsa van, akkor vigyük le $x.c_i$ -be egy kulcsot x -ből, és az $x.c_i$ közvetlen bal vagy jobboldali testvérétől vigyük fel egy kulcsot x -be, és vigyük át a megfelelő gyerek mutatóját a testvértől $x.c_i$ -be
 - b. Ha $x.c_i$ -nek, és (mindkét) közvetlen testvérének $t - 1$ kulcsa van, akkor egyesítsük $x.c_i$ -t az egyik testvérével, majd vigyük le egy kulcsot x -ből ebbe az egyesített csúcsba, középre

- 3.b: D törlése



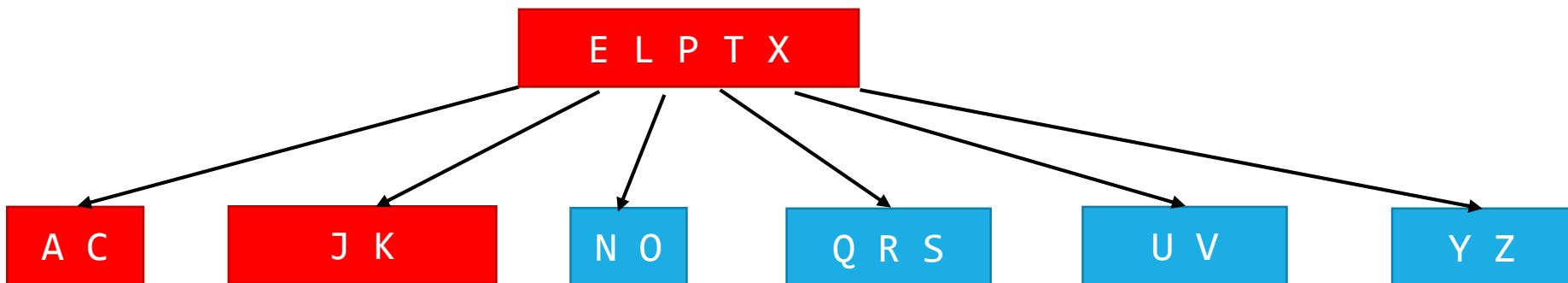
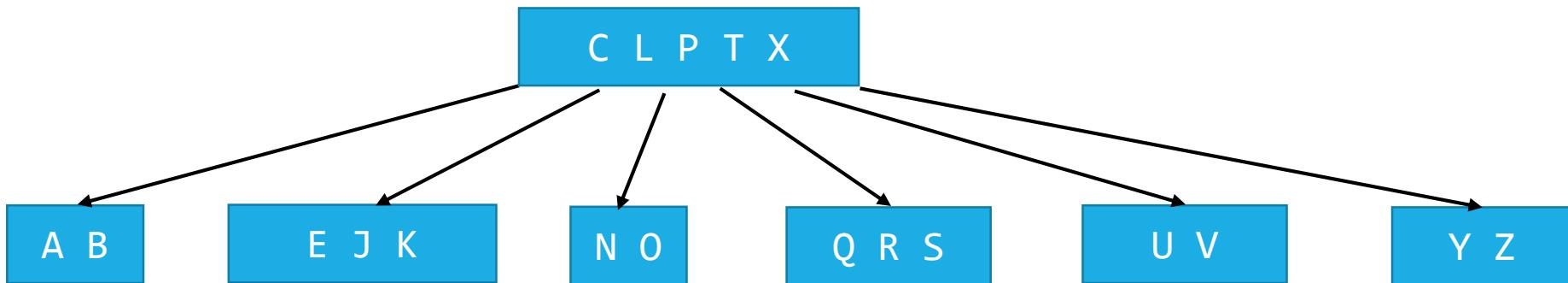
A B-fa műveletei – Törlés

- 3.b: D törlése
 - A fa magassága csökkent



A B-fa műveletei – Törlés

- 3.a: B törlése





A B-fa műveletei – Törlés

B-Tree-Delete(x, k)

```
//k a törlendő kulcs
//x a részfa gyökere, amiből k-t törölni szeretnénk
//B-Tree-Delete igazat ad vissza, ha sikerült
//Feltételezi, hogy x-nek legalább t kulcsa van
```

```
if x is a leaf then      //ha levél
  if k is in x then
    törlök k-t x-ból
    return true;
  else return false        //k nem részfa
else                      //x egy belső csúcs
```



A B-fa műveletei – Törlés

```
if k is in x then
    y = az x k-t megelőző gyereke
    if y-nak van legalább t kulcsa then
        k' = k megelőzője
        másoljuk át k'-t k-ba
        B-Tree-Delete(y, k')           // rekurzív hívás
    else //y -nak t-1 kulcsa van
        z = az x k-t követő gyereke
        if z -nek van legalább t kulcsa then
            k' = a k rákövetkezője
            másoljuk át k' -t k-ba
            B-Tree-Delete(z, k')       // rekurzív hívás
        else //y-nak is és z-nek is t-1 kulcsa van
            vonjuk össze k-t és a teljes z-t y-ba ->
            -> y-nak most 2t-1 kulcsa lesz
            k-t és a z-re mutató pointert töröljük x-ből.
            B-Tree-Delete(y, k)         // rekurzív hívás
```



A B-fa műveletei – Törlés

else //k nem belső csúcsa x-nek

c_i[x] mutat annak a részfának a c gyökerére, ami tartalmazhatja a k-t

if c-nek t-1 kulcsa van **then**

if c -nek van olyan közvetlen bal/jobb testvére (z), aminek
t vagy több kulcsa van **then**

Legyen k1 a kulcs x-ben, ami megelőzi/követi c-t

Vidd k1-t c-be mint első/utolsó kulcsot

Legyen k2 az első/utolsó kulcs a z közvetlen bal/jobb testvérben

Helyettesítsd k1-t x-be k2-vel z-ből (vidd fel k2-t x-be).

Vidd a z utolsó/első gyerek részfáját a c első/utolsó gyerek
részfájának

else

//c-nek és mindkét közvetlen testvérének t-1 kulcsa van

// összevonjuk c-t az egyik közvetlen testvérével és

// x megf. kulcsát középre tesszük

// (Ez új gyökérhez vezethet)

B-Tree-Delete(c, k)



Pázmány Péter Katolikus Egyetem
Információs Technológiai és Bionikai Kar

Java tantárgy tavasszal¹

Következő