ADATSZERKEZETEK ÉS ALGORITMUSOK

B-fa

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 (blokkot)
    - A fa csúcsai legyenek blokkok, a költség a blokkelérések száma

B-fák

- Háttértár műveletek modellezése:
  - Legyen **x** egy objektumra mutató pointer.
  - Ha az objektum pillanatnyilag a központi memóriában van, akkor a mezőire a szokásos módon hivatkozhatunk – x.kulcs
  - Ha a mágneslemezen van, akkor először a `LEMEZROL_OLVAS(x)` kell, ami beolvassa az x által hivatkozott objektumot a központi memóriába, utána lehet csak a mezőire hivatkozni.
  - Hasonlóan a `LEMEZRE_IR(x)` menti el a megváltozott mezőjű (x által hivatkozott) objektumot a mágneslemezre.
  - (Feltesszük, hogy ha x már a memóriában van, akkor `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 ← az objektum mutatója
LEMEZROL_OLVAS(x)
x mezőit olvasó és módosító műveletek
LEMEZRE_IR(x)
```

  - kimarad, ha x egyik mezője sem változott meg

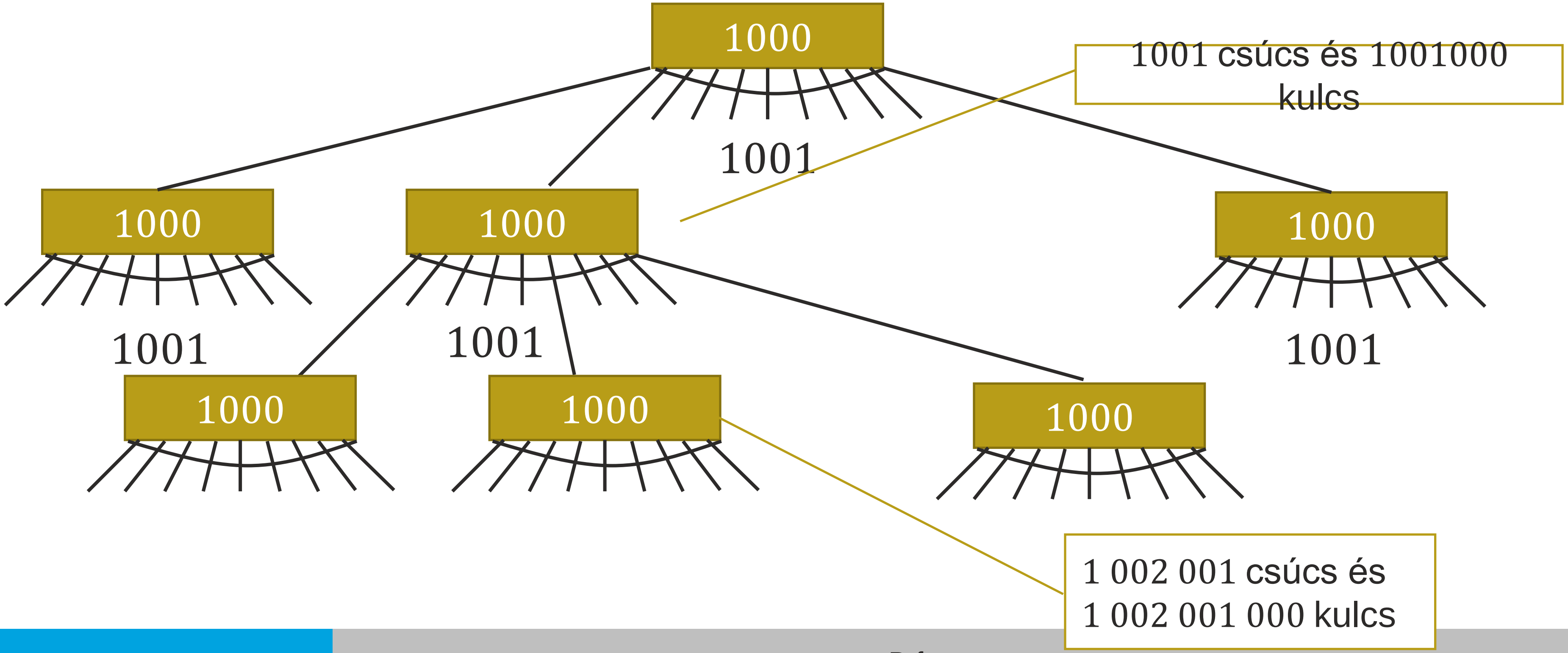
```
további x mezőit olvasó műveletek
…
```

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 gyermeke van mindegyiknek



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 információ
    - 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:
     - $x.kulcs_1 < x.kulcs_2 < \cdots < x.kulcs_{n[x]}$
     - $x.level$ – logikai változó, IGAZ, ha x levél, HAMIS, ha x egy belső csúcs
  2. Ha x egy belső csúcs (nem levél), akkor tartalmazza a $x.c_1, x.c_2, \ldots\ x.c_{n[x]+1}$ mutatókat az x gyerekeire
     - A levél csúcsoknak nincsenek gyerekeik, a levelek $x.c_i$ mutatói definiálatlanok

B-fák – Definíció

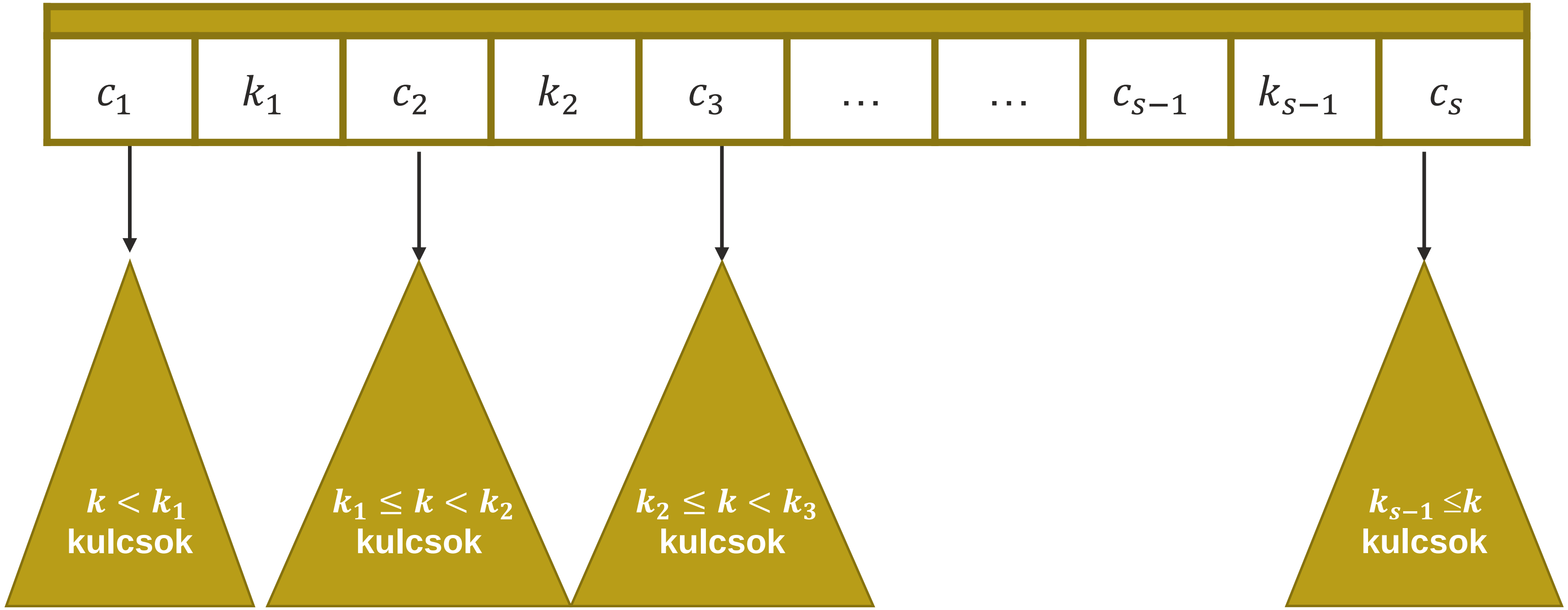
3. Az $x.kulcs_i$ értékek meghatározzák a kulcsértékeknek azokat a tartományait, amelyekbe a részfák kulcsai esnek
$$k_1 \le x.kulcs_1 < k_2 \le x.kulcs_2 < \cdots \le x.kulcs_{n[x]} < k_{n[x]+1}$$
4. Minden levélnek azonos a mélysége, ez az érték a fa h magassága
5. A csúcsokban tárolható kulcsok darabszámára adott egy alsó és egy felső korlát. Ezeket a korlátokat egy $t \ge 2$ 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 mindig 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

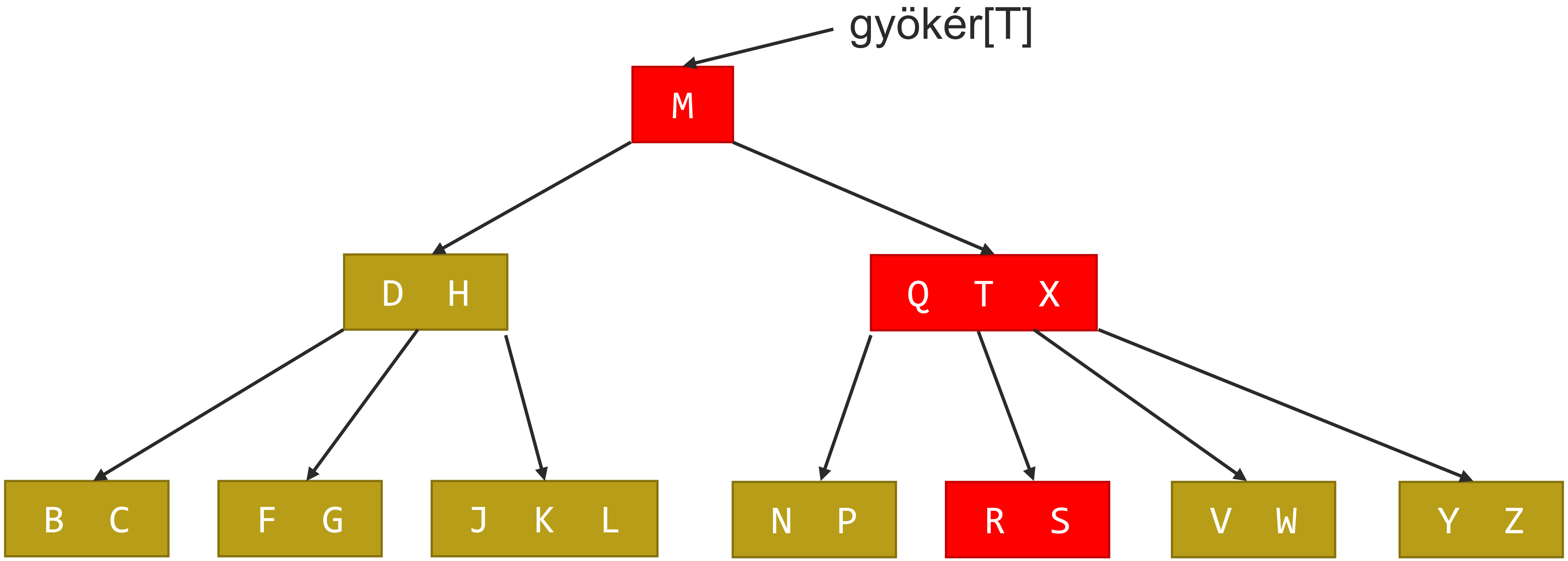
A B-fa műveletei – Keresés

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

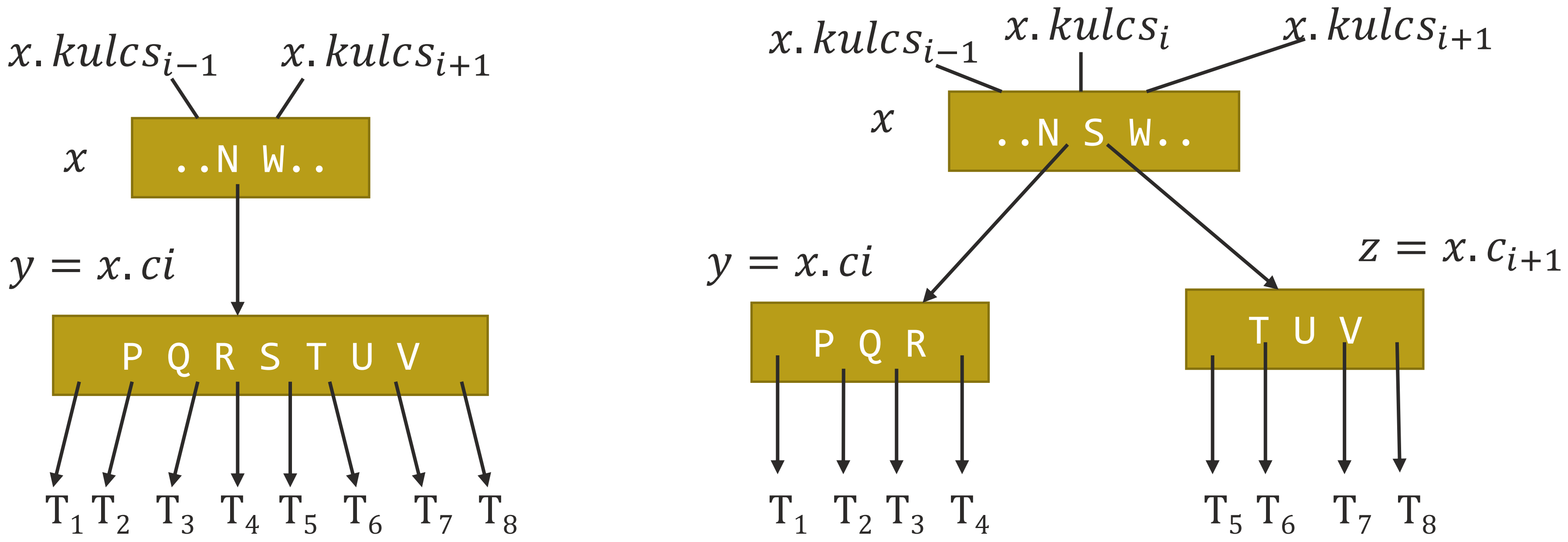
- `B-FAT-LETREHOZ` – egy üres gyökércsúcsot ad.
  - Használja a `PONTOT-ELHELYEZ` eljárást, ez $\mathcal{O}(1)$ idő alatt lefoglalja az új csúcsnak a tároló egy blokkját.
    - Feltételezzük, hogy nincs szüksége a `LEMEZROL-OLVAS` eljárás meghívására

```
B-FAT-LETREHOZ(T)
  x← PONTOT-ELHELYEZ()
  x.levél←IGAZ
  n[x]←0
  LEMEZRE-IR(x)
  gyökér[T]←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←PONTOT-ELHELYEZ()
  z.levél← y.levél
  n[z]←t-1
  for j←1 to t-1 do
    z.kulcs_j←y.kulcs_j+t
  if not y.levél
    then for j←1 to t do
              z.c_j←y.c_j+t
  n[y]←t-1
```

- $\mathcal{O}(1)$ idő alatt lefoglalja az új csúcsnak a tároló egy blokkját
- Átmásoljuk bele az y „végét”
- Most már y mérete is $t - 1$

A B-fa műveletei – Csúcs szétvágása

```
for j←n[x]+1 downto i+1 do
  x.c_{j+1}←x.c_j
x.c_{i+1}←z
for j←n[x] downto i do
  x.kulcs_{j+1}←x.kulcs_j
x.kulcs_{i+1}←y.kulcs_t
n[x]←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 egymenetes, a fában lefelé haladó algoritmussal oldható meg, a végrehajtáshoz $\mathcal{O}(h)$ tárolóhozzáférés kell
  - A szükséges központi egység idő $\mathcal{O}(t * h) = \mathcal{O}(t \log_t n)$

```
B-FABA-BESZUR(T,k)
  r←gyökér[T]
  if n[r]=2t-1
    then s←PONTOT-ELHELYEZ()
      gyökér[T] ← s
      s.levél← HAMIS
      n[s] ← 0
      s.c1 ← r
      B-FA-VÁGÁS-GYEREK(s,1,r)
      NEM-TELITETT-B-FABA-BESZUR(s,k)
    else NEM-TELITETT-B-FABA-BESZUR(r,k)
```

ha telített a gyökércsúcs, vág

A B-fa műveletei – Beszúrás

```
NEM-TELITETT-B-FABA-BESZUR(x,k)
  i←n[x]
  if x.levél
    then
       while i≥1 és k<x.kulcs_i do
         x.kulcs_i+1←x.kulcs_i
         i←i-1
       x.kulcs_i+1←k
       n[x]←n[x]+1
       LEMEZRE-IR(x)
    else
       while i≥1 és k<x.kulcs_i do
         i←i-1
       i←i+1
       LEMEZROL-OLVAS(x.c_i)
  …
```

hátulról kezdjük

itt a k helye
x mérete nő

ha nem levél,
megkeressük a helyét

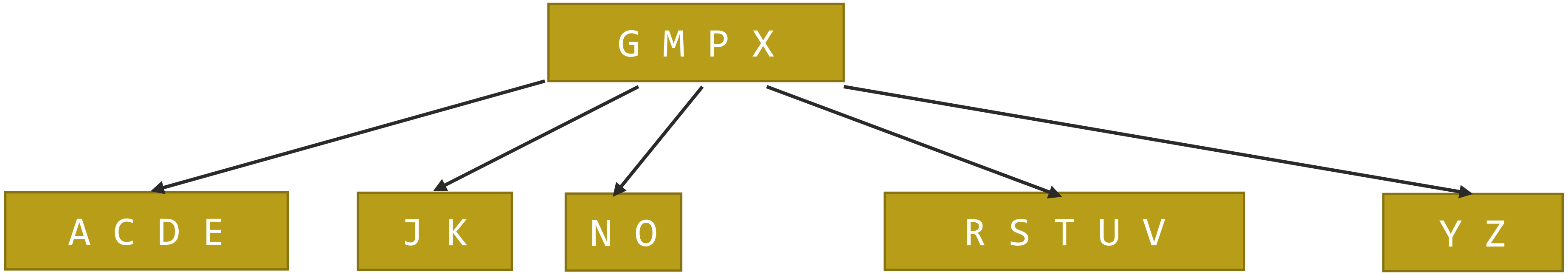
A B-fa műveletei – Beszúrás

```
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
  NEM-TELITETT-B-FABA-BESZUR(x.ci,k)
```

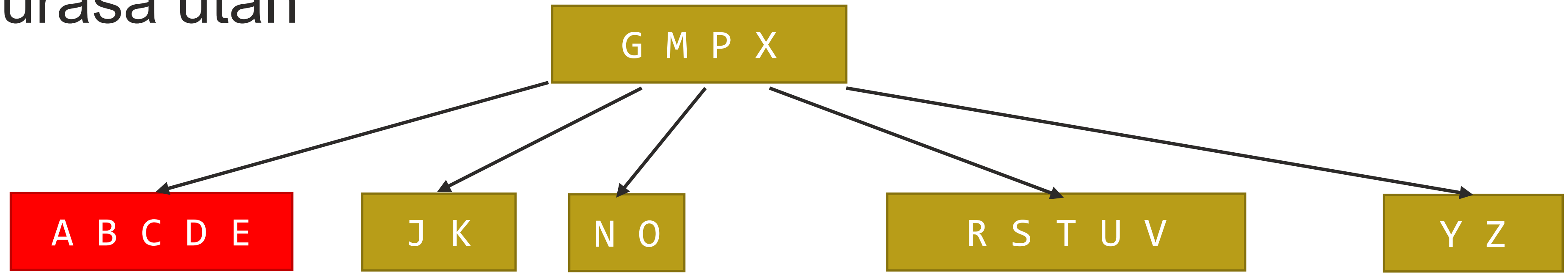
Telített?

A B-fa műveletei – Beszúrás

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

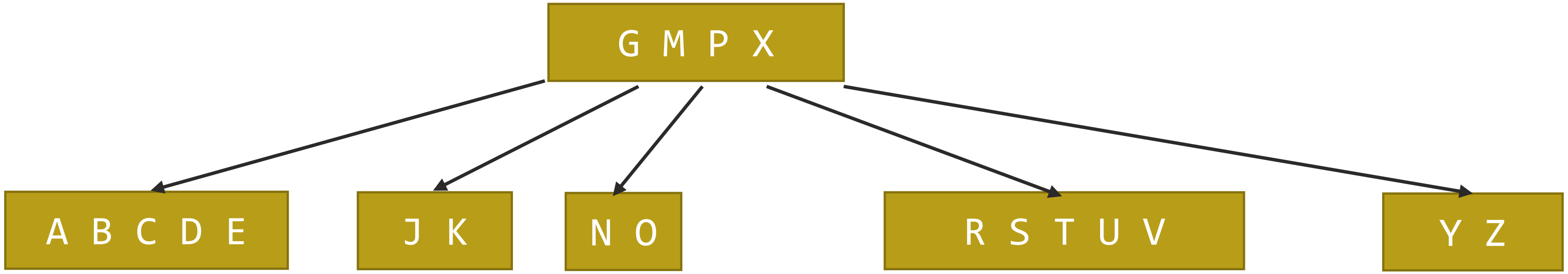


- B beszúrása után

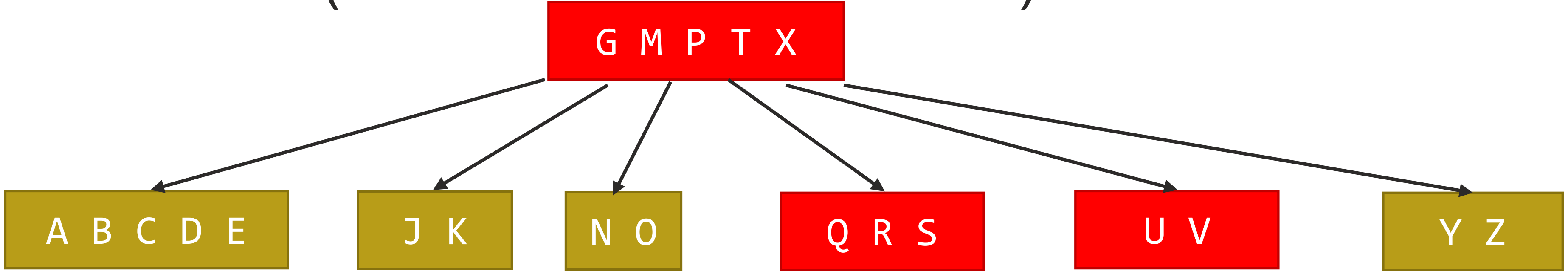


A B-fa műveletei – Beszúrás

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

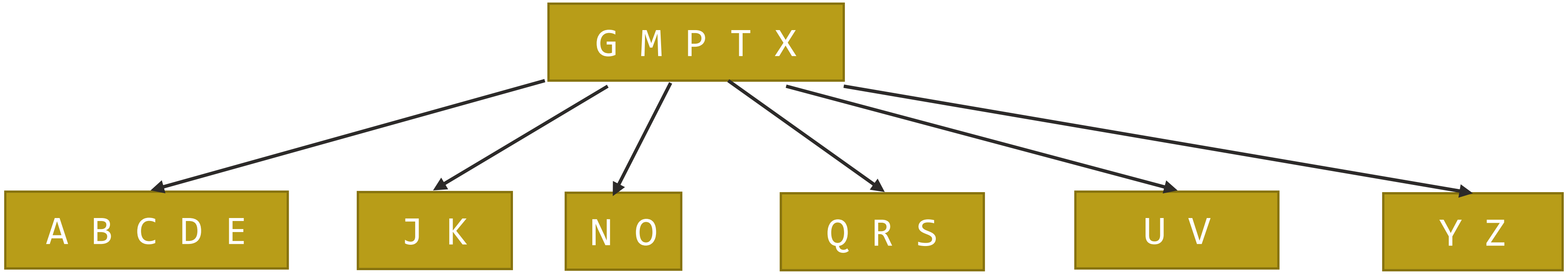


- Q beszúrása után (Az RSTUV csúcs telített)

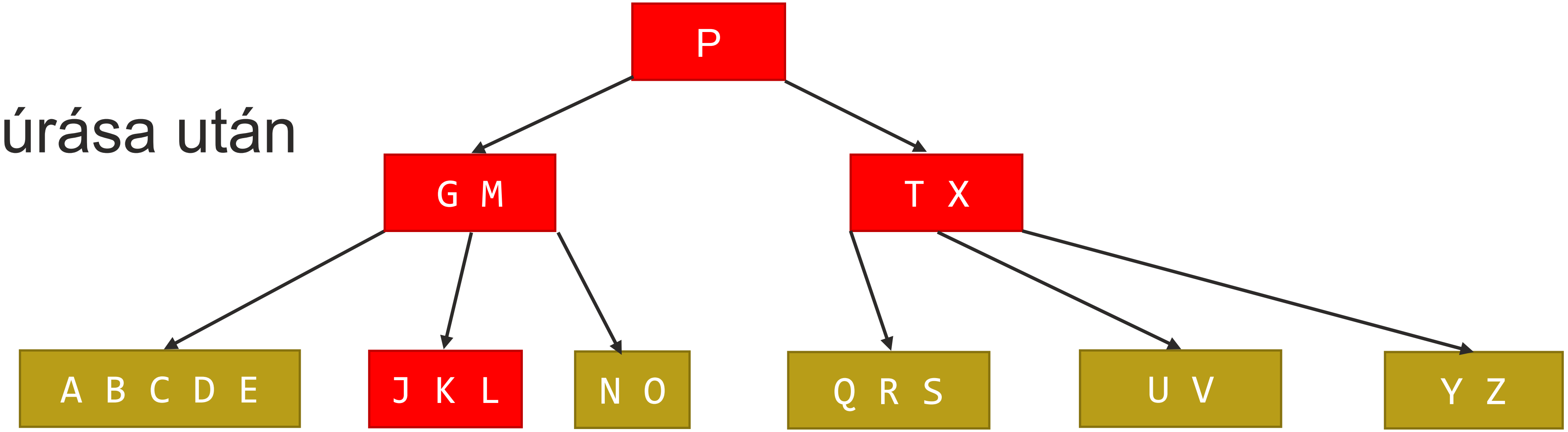


B-fa műveletei – Beszúrás

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

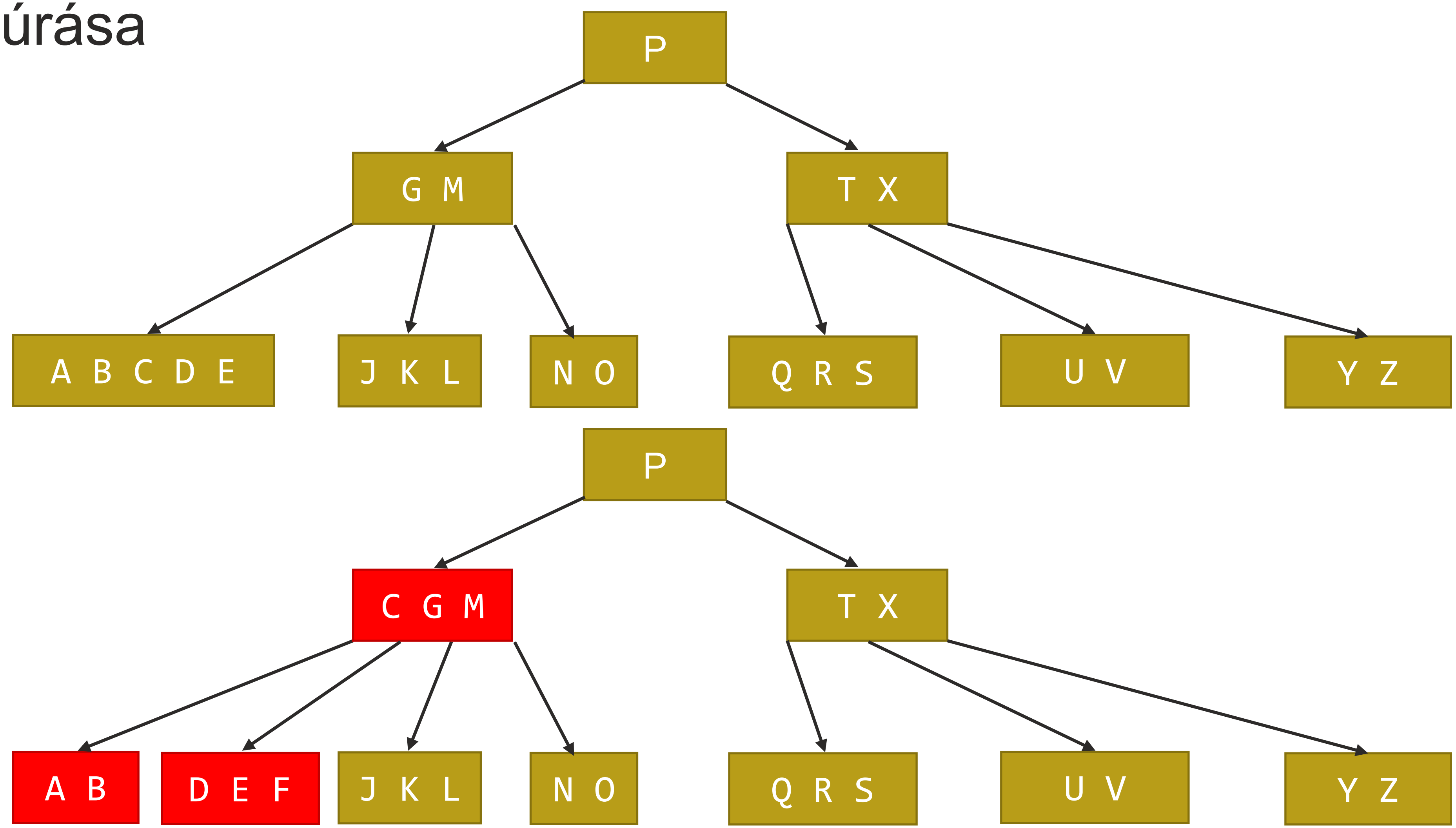


- L beszúrása után



B-fa műveletei – Beszúrás

- F beszúrása

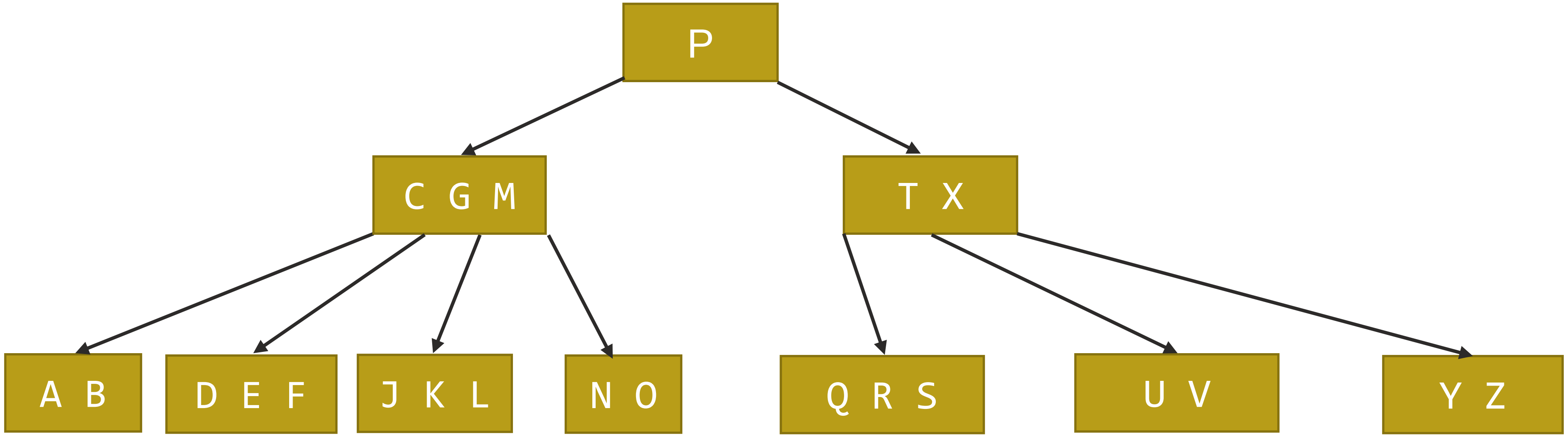


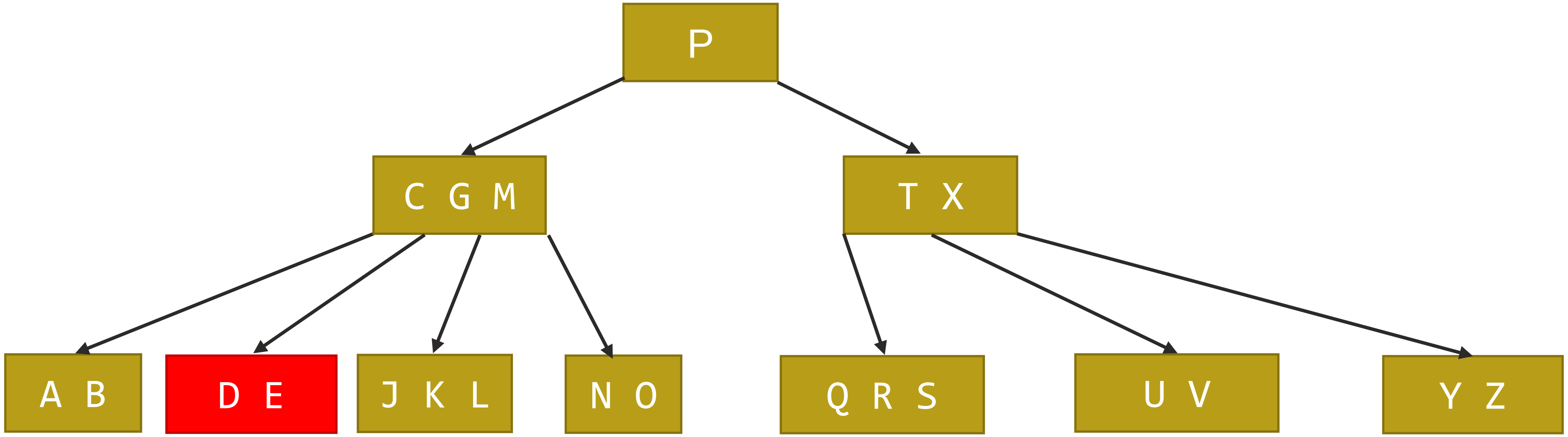
A B-fa műveletei – Törlés

- Törlés – kulcsot nemcsak levélből, hanem tetszőleges csúcsból lehet törölni.
  - 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 töröljük az x-ből

A B-fa műveletei – Törlés

- F törlése





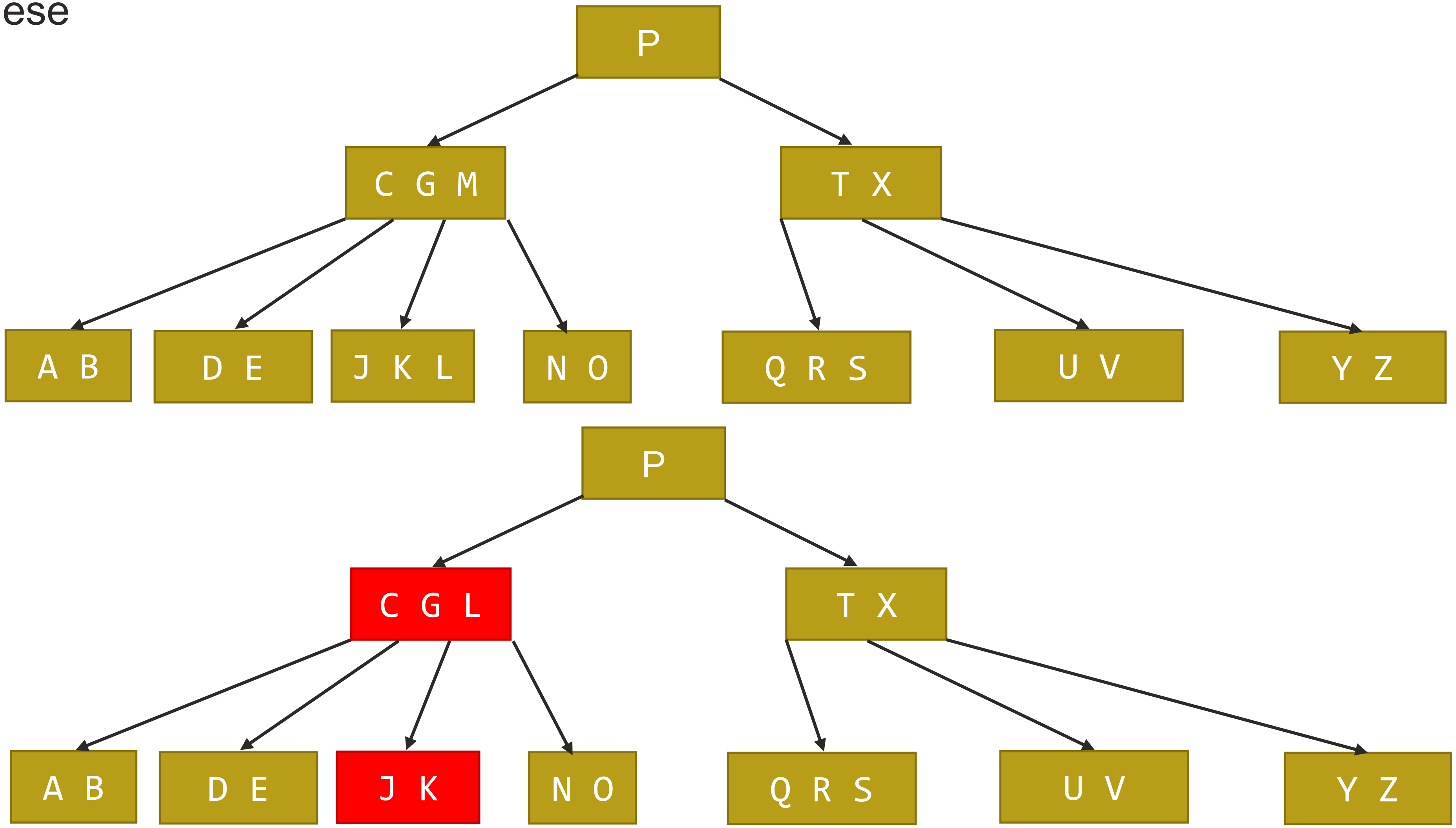
A B-fa műveletei – Törlés

- 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.

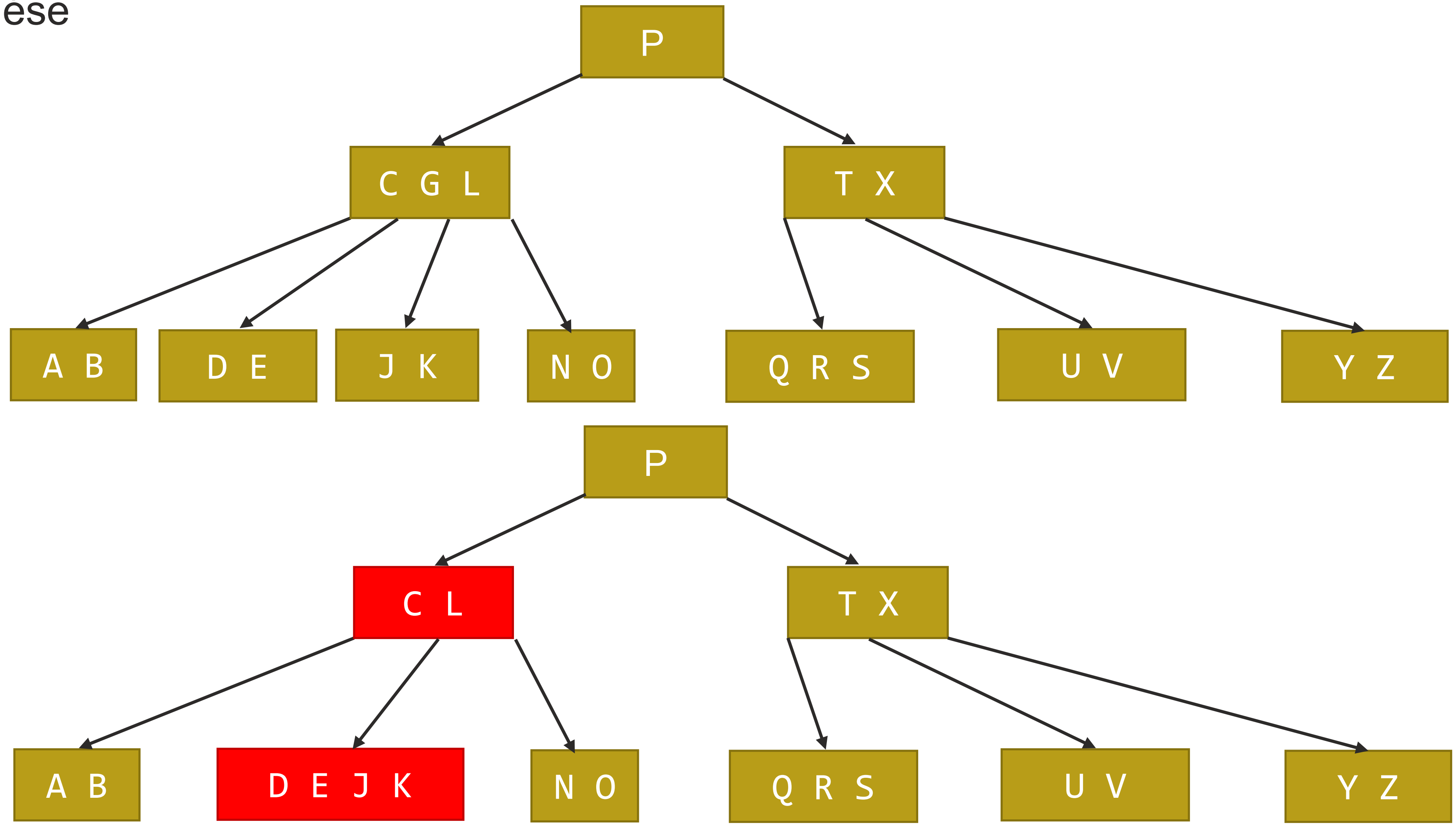
A B-fa műveletei – Törlés

- 2.a: M törlése



A B-fa műveletei – Törlés

- 2.c: G törlése

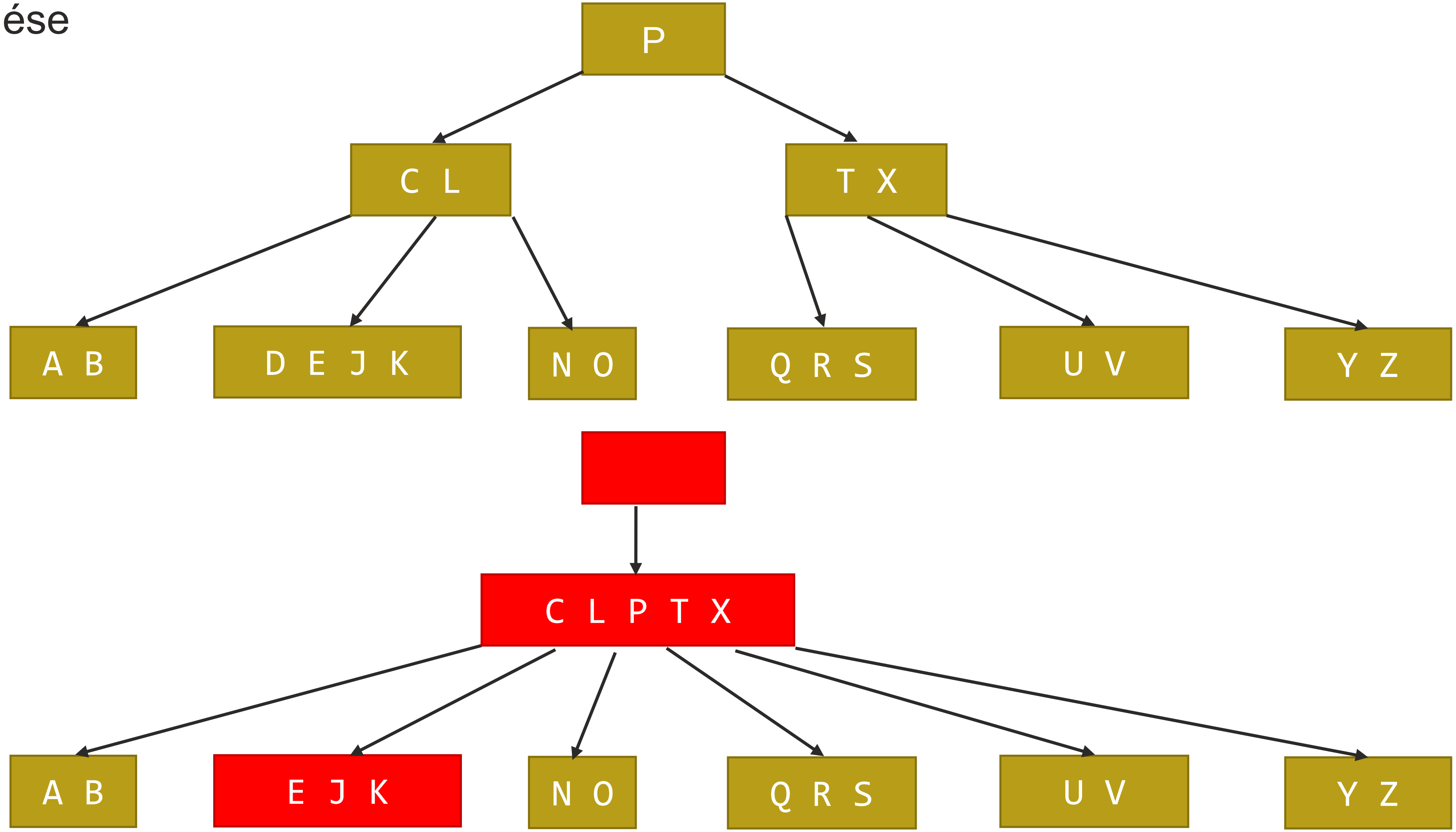


A B-fa műveletei – Törlés

- Lehetőségek
  3. Ha a k kulcs nincs benne az x belső csúcsban, akkor határozzuk meg annak a részfának az $x.c_i$ gyökércsúcsát, amelyikben benne lehet a k, ha egyáltalán szerepel.
     Ha $x.c_i$ –nek csak $t-1$ csúcsa van, akkor a 3a vagy 3b szerint járjunk 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 vigyünk le $x.c_i$–be egy kulcsot
        x-ből, és az $x.c_i$ közvetlen bal vagy jobboldali testvérétől vigyünk 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 vigyünk le egy kulcsot x-ből ebbe az egyesített csúcsba, középre

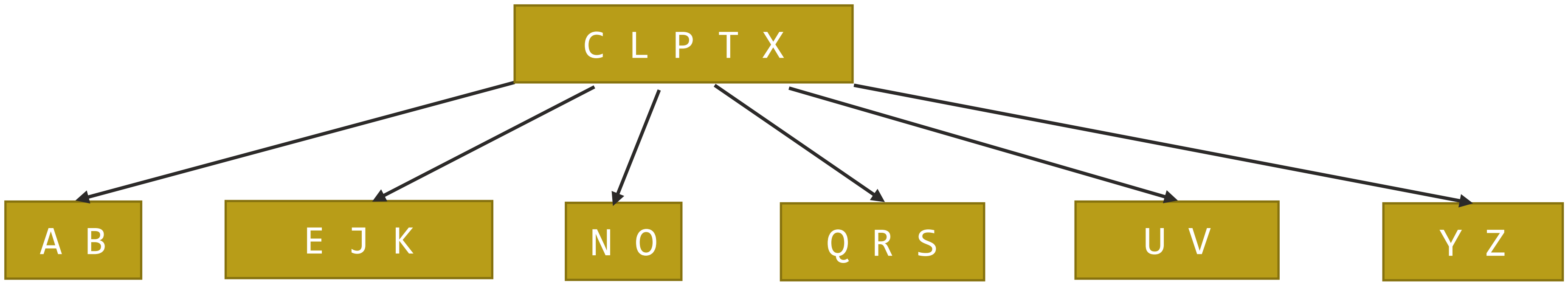
A B-fa műveletei – Törlés

- 3.b: D törlése



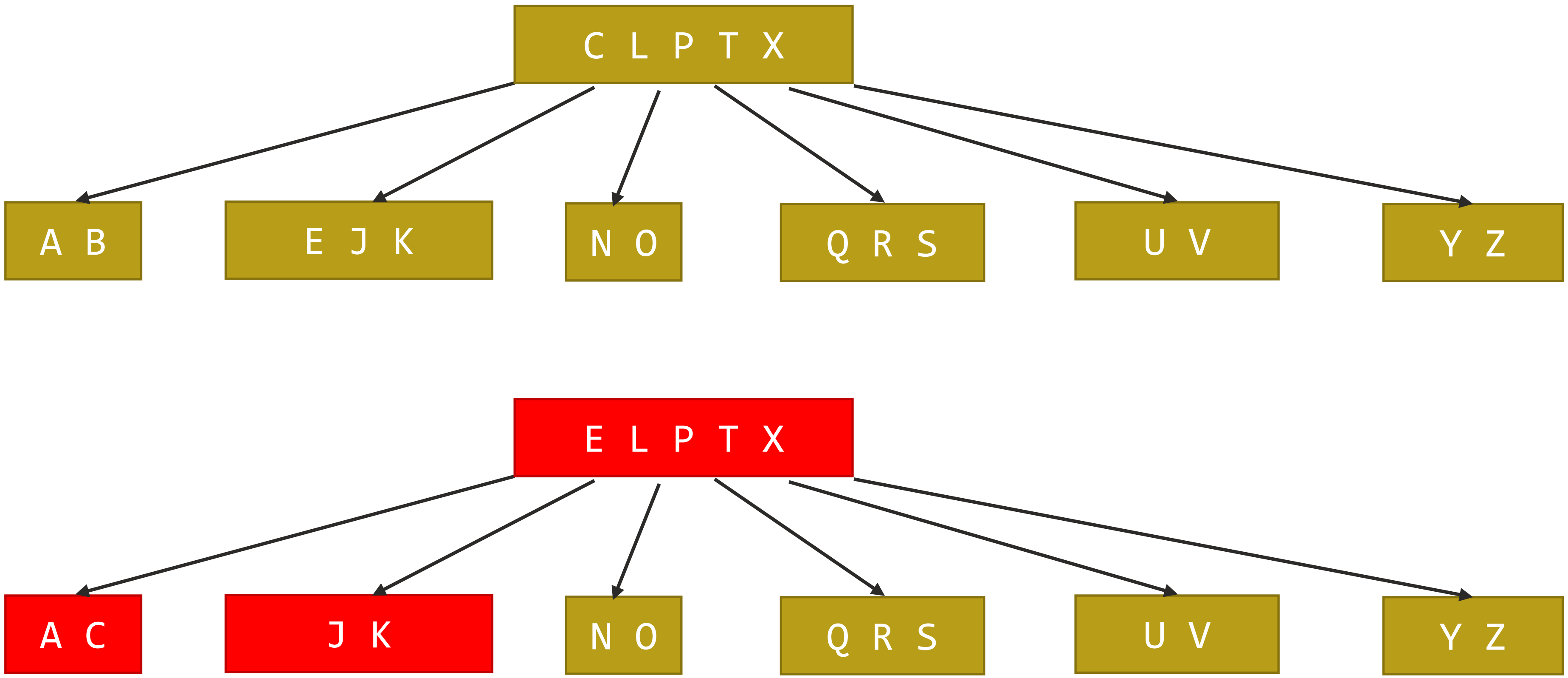
A B-fa műveletei – Törlés

- 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
    töröld k-t x-ből
    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
  ci[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)
```

B-fa családfa

B, B*, B+

- B fa (https://github.com/torvalds/linux/ https://www.cs.usfca.edu/~galles/visualization/BTree.html)
  - Egy kulcs és a kulcshoz tartozó rekord adatai együtt vannak, levélben, belső csúcsban egyaránt
- B+ fa (https://www.cs.usfca.edu/~galles/visualization/BPlusTree.html)
  - A rekordadatok csak levelekben találhatók
  - Belső csúcsban kulcsok (másolatai)
  - 2-3 fához hasonlóan
- B* fa
  - Nagyobb kiegyensúlyozottságra törekszik belső csúcsok esetén
  - A gyökérnél ⅔-os telítettség a cél az ½ helyett
  - A beszúrási lépésnél törekszik a csúcsvágás elkerülésére, amíg lehet
  - Komplexebb törlési algoritmus

Használat

- Fájlrendszerek esetén
  - B*: HFS, Reiser4
  - B+: XFS
  - B: HFS+, NTFS (variáns), JFS2, BTRFS, ext3, ext4 (Htree)
- Adatbázisok esetén
  - MariaDB
    - engine függő, de a B-fa általában elérhető indexer
  - Postgres
  - MySQL
  - Oracle
  - …

Elméleti ZH / Vizsga

Következő téma