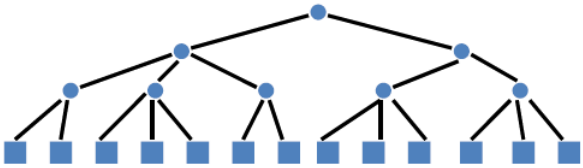
10. tétel

Adatszerkezetek és algoritmusok vizsga

Frissült: 2013. január 28.

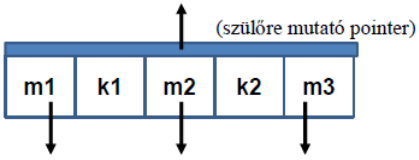
2-3 fák



Hatékony keresőfa-konstrukció. Ez is fa, de a binárisnál annyival bonyolultabb hogy egy nem-levél csúcsnak 2 vagy 3 fia lehet. Lefelé irányított gyökeres fa, melyre igaz hogy:

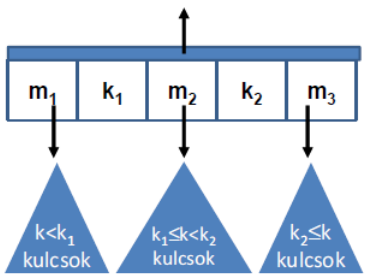
1. A rekordok a fa leveleiben helyezkednek el, a kulcs értéke szerint balról jobbra növekvő sorrendben. Egy levél egy rekordot tartalmaz.
2. Minden belső (azaz nem levél) csúcsból 2 vagy 3 él megy lefelé; ennek megfelelően a belső csúcsok egy, illetve két $k \in U$ kulcsot tartalmaznak.

A belső pontok szerkezete:



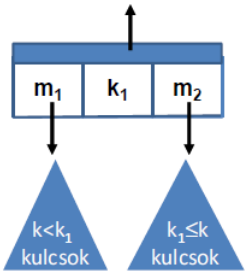
A belső csúcs kétféle lehet:

- 3 gyerekes



- m_1, m_2, m_3 mutatók a csúcs *részfáira* mutatnak.
- k_1, k_2 U-beli *kulcsok*, melyekre $k_1 < k_2$.
- 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.
- m_3 részfájában k_2 a legkisebb kulcs.

- 2 gyerekes



- m_1, m_2 mutatók a csúcs *részfáira* mutatnak.
- k_1 U-beli *kulcs*.
- 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.

Megjegyzés. Ha nincs elem a fában ($n = 0$), $t = NIL$ vagy üres a gyökér. Ha 1 elem van ($n = 1$), akkor itt kivételesen egy gyereke van a gyökérnek.

A fa magassága korlátos: $2^h < n < 3^h \Rightarrow h \leq \log_2 n$ (még 2-es elágazási tényező esetén is!)

Műveletek

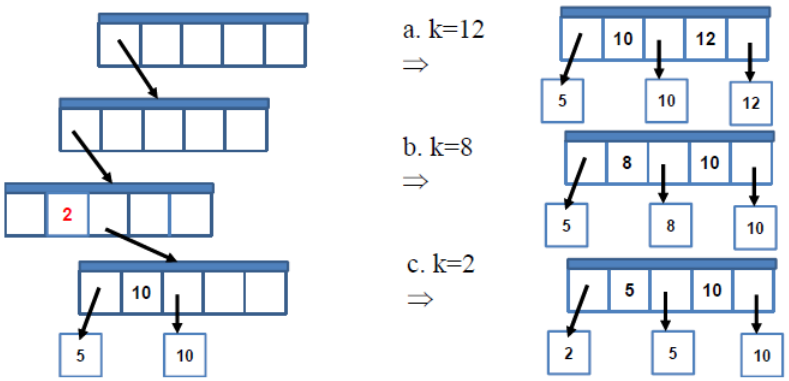
Keresés

gyorsasága: $\theta(\log_2 n)$

Beszúrás

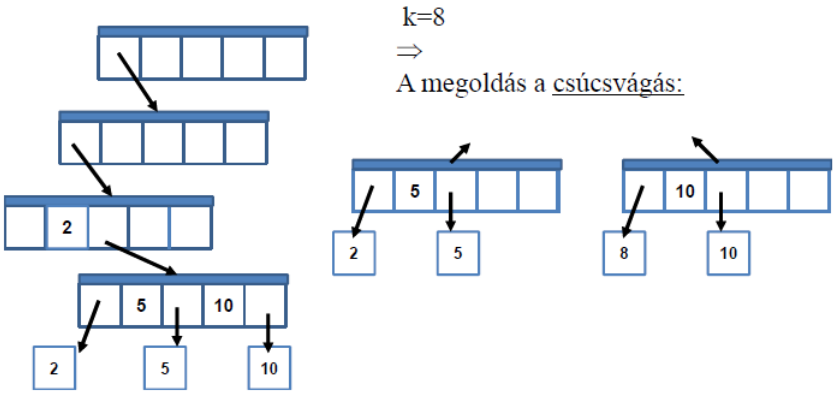
Először is kereséssel meg kell határozni a helyét az új elemnek. Két eset lehet, attól függően, hogy a legalsó belső pontnak hány gyereke van: 2 vagy 3.

Ha *2 gyereke van*, akkor elfér még egy 3. is.

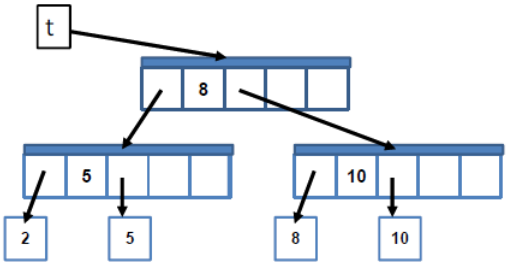


Felfelé haladva korrigálni kell a megfelelő kulcsot 2-re. (ha a nagyszülőben az 5-nél kisebb nem volt, egy leágazásos elem nem szúrható be.)

Ha *3 gyereke van*, itt már nem lehet közé szúrni, *csúcsvágás*t kell végezni.



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.

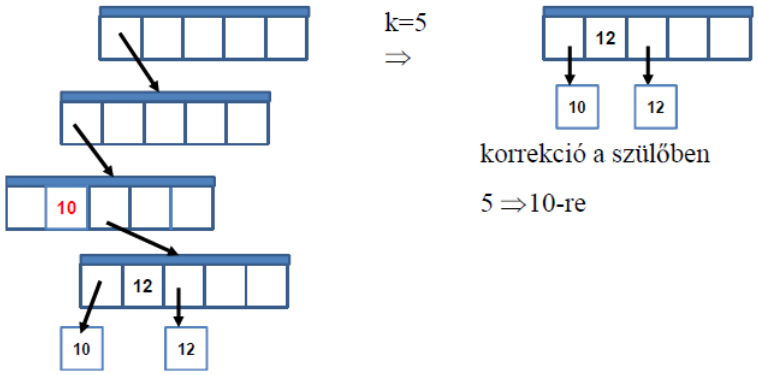


Nőtt a magasság (h)!

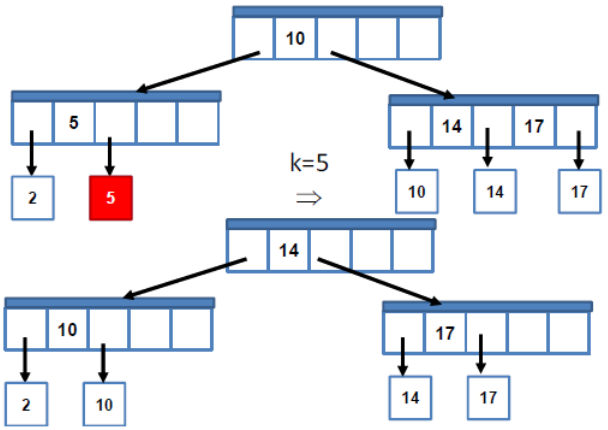
Törlés

Először megkeressük a törlendő csúcsot. Itt is, több esetre osztjuk szét a feladatot, most három eset fordul elő:

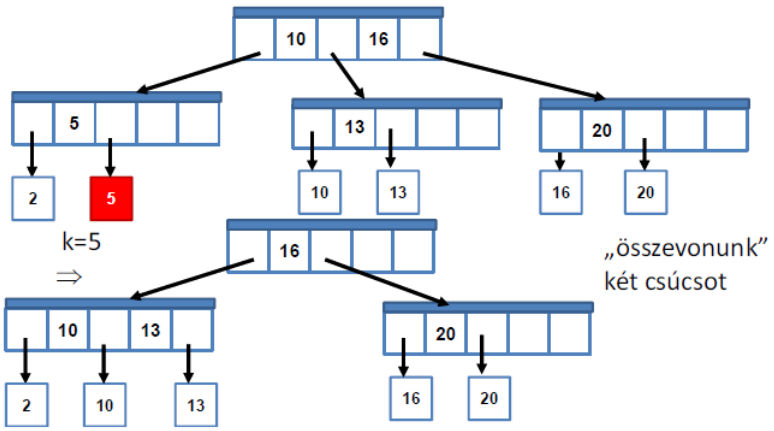
Ha a szülőnek *3 gyereke van*, nincs nagy baj, 2-re csökken:

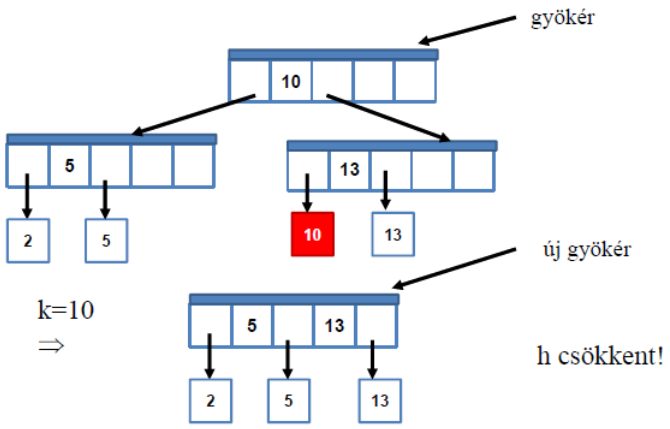


Ha viszont 2 gyereke volt, akkor 1 testvére van, ha viszont van 3 gyerekes testvére, az átad 1 gyereket neki.



Ha viszont még az sincs, össze kell vonni a testvér csúcsait, és utána feljebb folytatni, egész a gyökérig összevonni, ameddig kell.





Ha a gyökérig eljut, a magasság ilyenkor csökken.
A költség: $O(\log_2 n)$

B-fák

A B-fák a fenti 2-3 fák általánosításai. Nagy méretű adatbázisok, külső táron levő adatok feldolgozására használják. Sokféle szabványban felhasználták, lemezkezelésnél általánosan elterjedt. Az adatok kiolvasását szeretnénk felgyorsítani vele.

Definíció. Minden x csúcsnak a következő mezői vannak:

- a benne tárolt kulcsok darabszáma: $n[x]$ Ezekre adott egy *alsó* és egy *felső korlát*, ez a korlát $t \geq 2$, ő a B-fa *minimális fokszáma*.
- maguk a kulcsok, melyek nem csökkenő sorrendben vannak eltárolva:

$$x.kulcs_1 \leq x.kulcs_2 \leq ... \leq x.kulcs_{n[x]}$$

- Ha x *belső csúcs* (nem levél), akkor tartalmazza a

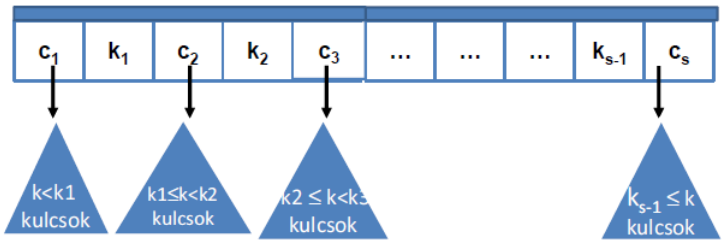
$$x.c_1, x.c_2, ...x.c_{n[x]+1}$$

mutatókat az ő *gyerekeire*. A c_i mutató olyan részfára mutat, amelynek kulcsai $kulcs_i$ és $kulcs_{i+1}$ közé esnek:

$$k_1 \leq x.kulcs_1 < k_2 \leq x.kulcs_2 < ... \leq x.kulcs_{n[x]} < k_{n[x]+1}$$

Az $x.kulcs_i$ érték meghatározzák a kulcsértékeknek azokat a tartományait, amelyekbe a részfák kulcsai esnek.

A belső csúcsok szerkezete:



- a *levél csúcs*oknak nincsenek leveleik, azok nincsenek is definiálva. Minden levélnek azonos a mélysége ami a fa h magassága.
- minden nem gyökér csúcsnak legalább $t-1$ kulcsa van. Így minden belső csúcsnak 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.

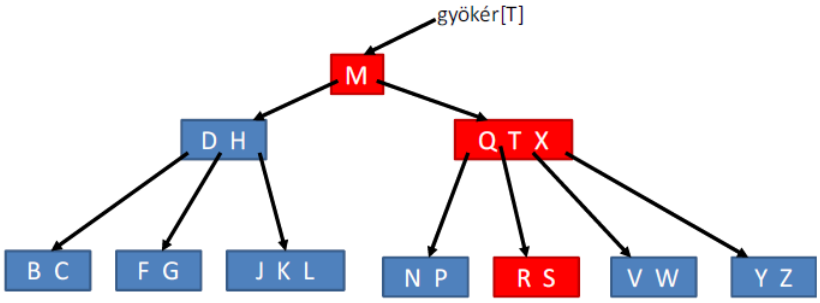
Műveletek

A 2-3 fákéhoz hasonló. A B-fa gyökere mindig a központi memóriában van, így a gyökércsúcsra LEMEZRŐL-OLVAS művelet nem kell, de LEMEZRE-ÍR kell akkor, ha a gyökércsúcs megváltozott. 2. Minden olyan csúcs, amely paraméterként szerepel, már a központi memóriában van, már végrehajtottunk rá egy LEMEZRŐL-OLVAS műveletet.

Keresés

x a részfa gyökércsúcsára mutató pointer, k a kulcs, amit ebben a részfában keresünk.

```
B-FÁBAN-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 LEMEZRŐL-OLVAS(x.c_i)
  return B-FÁBAN-KERES(x.c_i,k)
```



Ha az R betűt keressük, a pirosakon megyünk.

Műveletigény. Itt minden belső csúcsban $n[x] + 1$ lehetőséget kell megvizsgálni. Összes művelet idő: $\Theta(t * \log_t n)$

Létrehozás

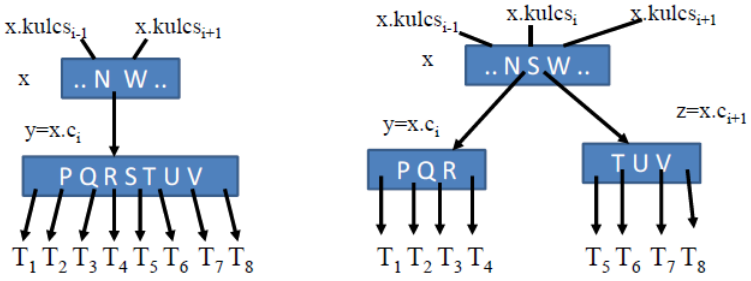
B-FÁT-LÉTREHOZ – egy üres gyökércsúcsot ad.

Használja a PONTOT-ELHELYEZ eljárást, ez $O(1)$ idő alatt lefoglalja az új csúcsnak a lemez egy lapját. Feltételezzük, hogy nincs szüksége a LEMEZRŐL-OLVAS eljárás meghívására.

```
B-FÁT-LÉTREHOZ(T)
 x←PONTOT-ELHELYEZ()
 x.levél←IGAZ
 n[x]←0
 LEMEZRE-ÍR(x)
 gyökér[T]←x
```

Ehhez $O(1)$ lemezművelet és $O(1)$ központi egység idő kell.

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.



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

```
B-FA-VÁGÁS-GYEREK(x, i, y)

z ← PONTOT-ELHELYEZ()      - O(1) idő alatt lefoglalja az új csúcsnak a lemez egy lapját
 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

 for j ← n[x]+1 downto i+1 do                    -- a csúcsokra mutatókat
   x.c _{j+1} ← x.c_j                             -- arrébb tesszük
  x.c_{i+1} ← z                                   -- z középre
 for j ← n[x] downto i do                        -- a kulcsokat arrébb tesszük
    x.kulcs_{j+1} ← x.kulcs_j
 x.kulcs_{i+1} ← y.kulcs_t                        -- a kulcs a helyére
 n[x] ← n[x] + 1                                  -- x elemszáma nőtt

 LEMEZRE-ÍR(y)
 LEMEZRE-ÍR(z)
 LEMEZRE-ÍR(x)
```

Beszúrás

Egy k kulcs beszúrása egy h magasságú T B-fába egy egymenetes, a fában lefelé haladó algoritmussal oldható meg, a végrehajtáshoz $O(h)$ lemezhozzáférés kell.

A szükséges központi egység idő $O(t * h) = O(t * \log_t n)$.

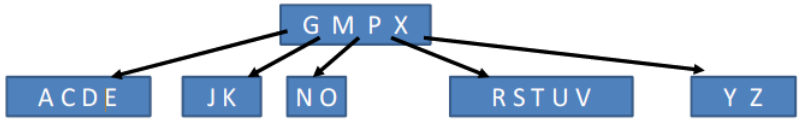
Pszeudokód:

```
B-FÁBA-BESZÚR(T,k)
 r ← gyökér[T]
 if n[r]=2t-1                    -- ha telített a gyökércsúcs, vág
  then s← PONTOT-ELHELYEZ()
    gyökér[T] ← s
    s.levél← HAMIS
    n[s] ← 0
    s.c_1 ← r
    B-FA-VÁGÁS-GYEREK(s,1,r)
    NEM-TELÍTETT-B-FÁBA-BESZÚR(s,k)
  else NEM-TELÍTETT-B-FÁBA-BESZÚR(r,k)
```

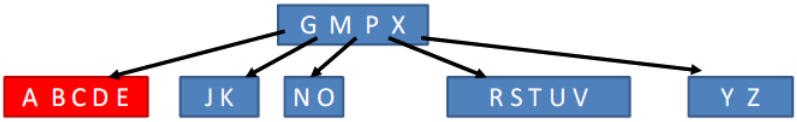
```
NEM-TELÍTETT-B-FÁBA-BESZÚR(x,k)
 i ← n[x]
 if x.levél
  then while i ≥ 1 és k< x.kulcs_i do
     x.kulcs_{i+1} ← x.kulcs_i
     i ← i-1
   x.kulcs_{i+1} ← k
   n[x] ← n[x] +1
   LEMEZRE-ÍR(x)
  else while i ≥ 1 és k < x.kulcs_i do
      i ← i-1
  i ← i+1
  LEMEZRŐL-OLVAS(x.c_i)

  if n[x.c_i]=2t-1              -- telített?
     then B-FA-VÁGÁS-GYEREK(x, i, x.c_i)
       if k > x.kulcs_i
         then i ← i+1
  NEM-TELÍTETT-B-FÁBA-BESZÚR(x.c_i,k)
```

Példa:

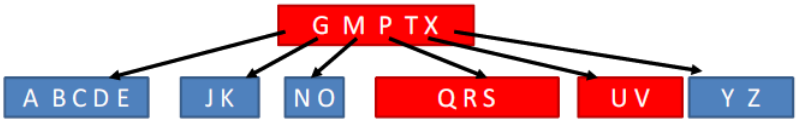
Tegyük fel hogy $t = 3$, azaz max. 5 kulcs lehet egy csúcsban. Most így néz ki a fánk:



B beszúrása után:

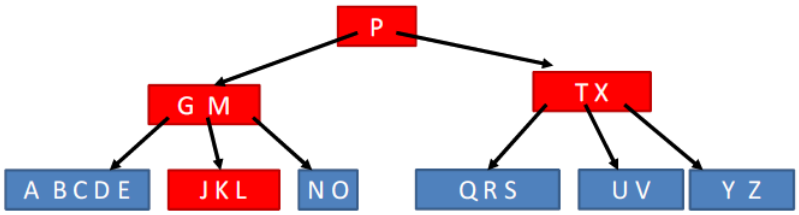


Q beszúrása után:

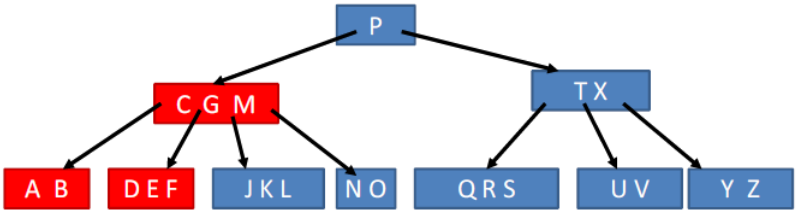


RSTUV telített volt!

L beszúrása után:



F beszúrása:



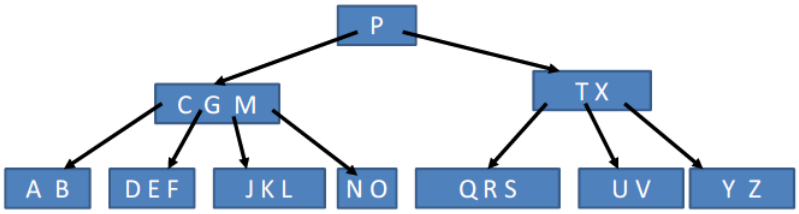
Törlés

Kulcsot nemcsak levélből, hanem tetszőleges csúcsból lehet törölni. Ügyelni kell arra, hogy a csúcs ne legyen túl kicsi (kivéve a gyökérben)

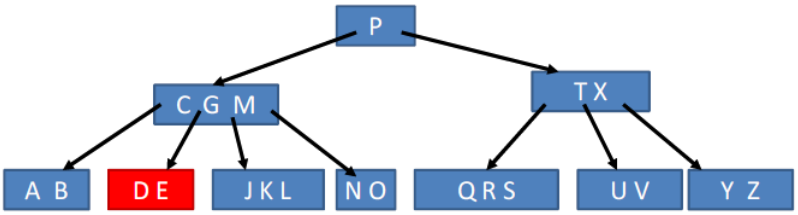
Lehetőségek.

1. a k kulcs az x csúcsban van, x egy levél, akkor a k kulcsot töröljük az x-ből.

   Példa



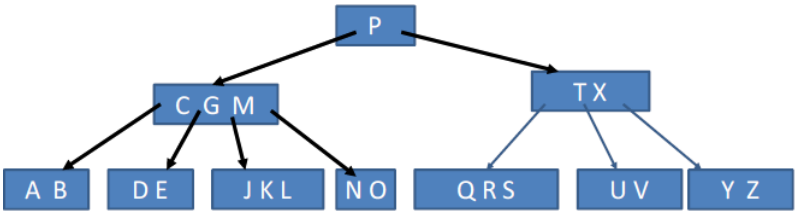
   Töröljük az F-t!



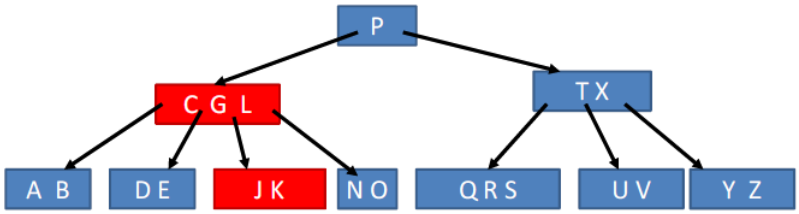
2. a k kulcs az x csúcsban van, x a fa egy belső csúcsa, akkor:

(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.

Példa. Eredeti fa:



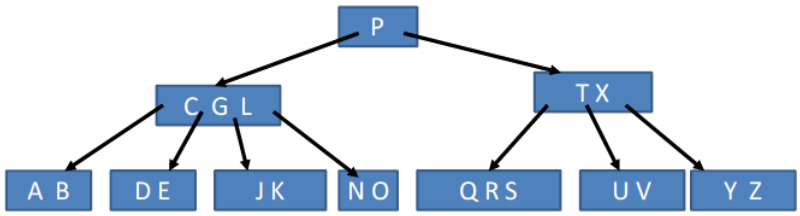
M törlése:



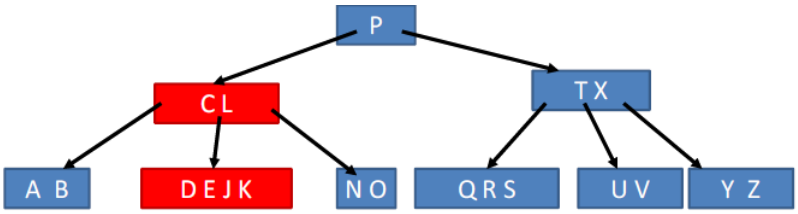
(b) szimmetrikusan, ha a z gyerek következik az x-beli k után, és z-nek legalább t kulcsa van, akkor keressük meg a z gyökércsúcsú részfában a k-t közvetlenül követő k' kulcsot. Rekurzívan töröljük k'-t és helyettesítsük k-t k'-vel az x-ben.

(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.

Példa. Eredeti fa:

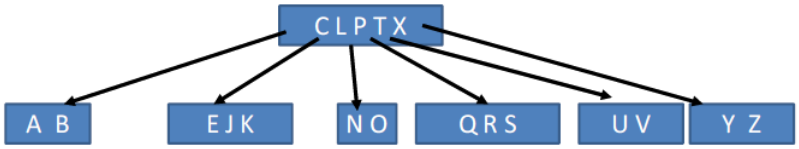


G törlése:

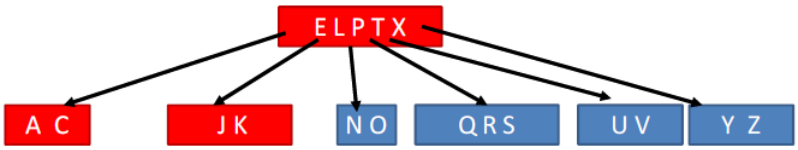


3. ha a k kulcs nincs benne az x belső csúcsban, akkor határozzuk meg annak a részfának az $x.c_i$ gyökércsúcsát, amelyikben benne lehet a k, ha egyáltalán szerepel. Ha $x.c_i$ –nek csak t-1 csúcsa van, akkor a 3a vagy 3b szerint járjunk el, mivel biztosítani kell, hogy annak a csúcsnak, amelyikre lépünk, legalább t csúcsa legyen. Ezután rekurzióval megyünk tovább.

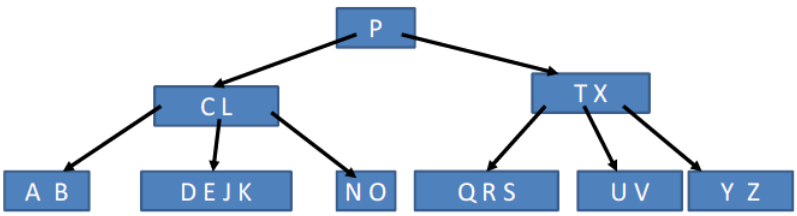
   (a) Ha $x.c_i$–nek csak t-1 csúcsa van, de van egy közvetlen testvére, amelyiknek legalább t csúcsa van, akkor vigyünk le $x.c_i$–be egy kulcsot x-ből, és a $x.c_i$ közvetlen bal vagy jobboldali testvérétől vigyünk fel egy kulcsot x-be, és vigyük át a megfelelő gyerek mutatóját a testvértől x.ci-be.
   Példa.



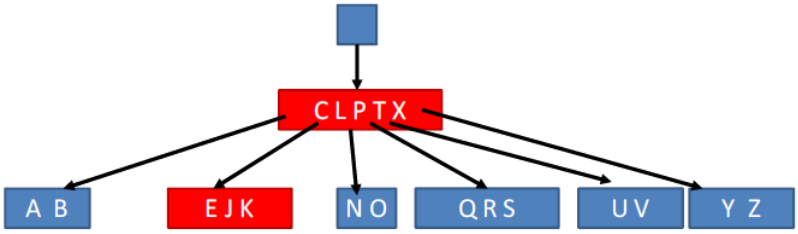
   B törlése:

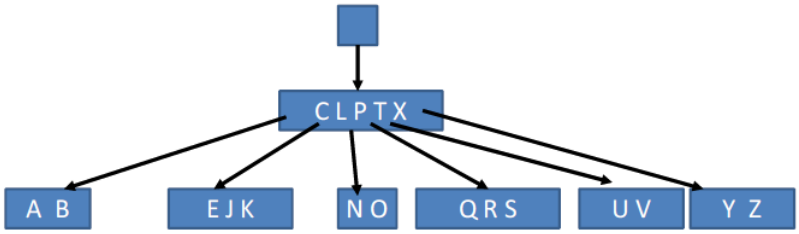


   (b) Ha $x.c_i$-nek, és mindkét közvetlen testvérének t-1 kulcsa van, akkor egyesítsük $x.c_i$-t az egyik testvérével, majd vigyünk le egy kulcsot x-ből ebbe az egyesített csúcsba, középre.
   Példa.

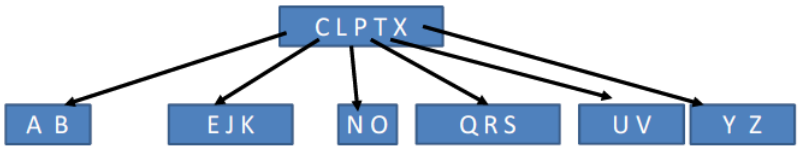


   D törlése:





a fa magassága csökken



Pszeudokód:

```
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
  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

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)
```