



Algoritmuselmélet 3. témakör

Pusztai Pál
pusztai@sze.hu

Tartalom

- Bináris keresőfák
 - Fabejárás, keresés, minimum, maximum, következő, beszúrás, törlés
- Piros-fekete fák
 - Forgató műveletek, beszúrás, javítás, törlés
- Adatszerkezetek kibővítése
 - Rendezettminta-fák
 - Kiválasztás, rang meghatározás
 - Intervallum-fák
 - Keresés
- B-fák
 - Tulajdonságok
 - Segéd eljárások
 - Keresés, beszúrás

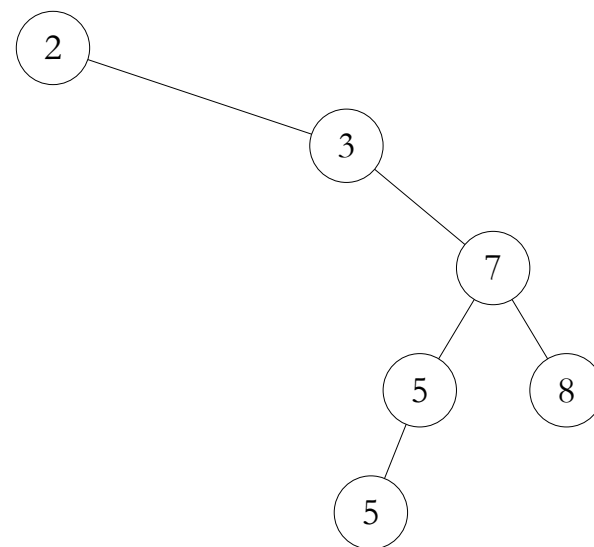
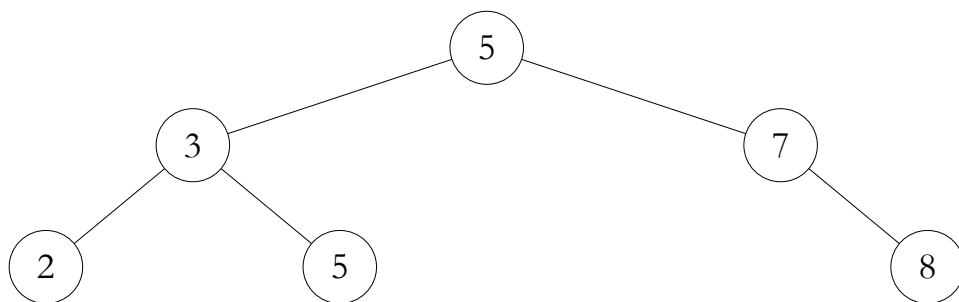
Bináris keresőfák

A **bináris fát** láncolt struktúraként ábrázolhatjuk, ahol minden csúcs egy önálló objektum.

A *kulcs* mező és a kísérő adatok mellett minden csúcs tartalmaz egy *bal*, egy *jobb* és egy *szülő* nevű mezőt is, amelyek rendre a csúcs bal- és jobboldali gyerekére, illetve a szülőjére mutatnak.

A **bináris-keresőfa tulajdonság**: Legyen x az adott bináris keresőfa egy tetszőleges csúcsa. Ha y az x baloldali részfájának egy csúcsa, akkor $kulcs[y] \leq kulcs[x]$, ha pedig y az x jobboldalára esik, akkor $kulcs[y] \geq kulcs[x]$.

A **keresőfák** olyan adatszerkezetek, amelyekre a dinamikus halmazok KERES, MINIMUM, MAXIMUM, ELŐZŐ, KÖVETKEZŐ, BESZÚR és TÖRÖL műveleteit értelmezzük, ezért mind **szótárként**, mind **elsőbbségi sorként** használhatók.



Bináris keresőfák

Bináris keresőfák

INORDER-FABEJÁRÁS(x)

```
1  if  $x \neq \text{NIL}$ 
2      INORDER-FABEJÁRÁS( $\text{bal}[x]$ )
3      Ki:  $\text{kulcs}[x]$ 
4      INORDER-FABEJÁRÁS( $\text{jobb}[x]$ )
```

Megjegyzés:

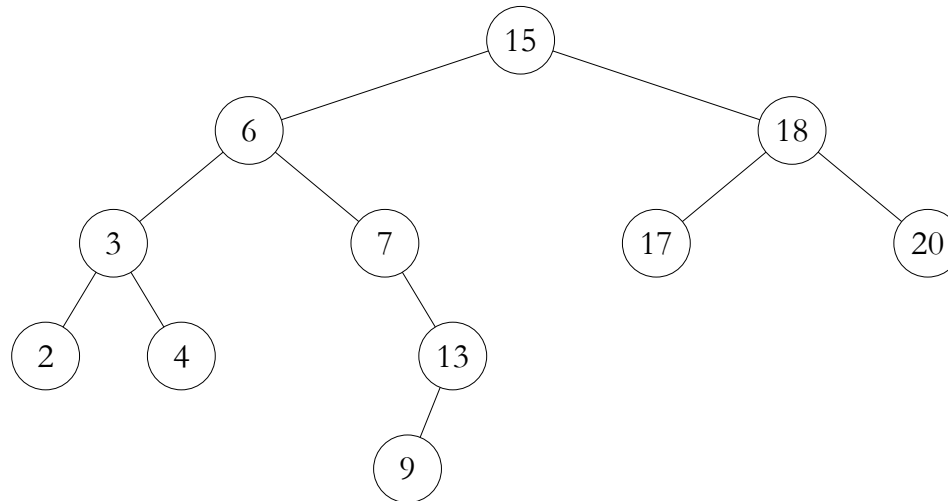
- Egy T bináris keresőfa összes elemének kiírása az INORDER-FABEJÁRÁS($\text{gyökér}[T]$) hívással történhet.
- Az **inorder bejárás** esetén a fa gyökerében lévő kulcsot a baloldali részfájában lévő értékek után és a jobboldali részfájában lévő értékek előtt (azok között) írjuk ki.
- A **preorder bejárás** esetén a gyökér kulcsát a részfáinak kulcsai előtt, míg a **posztorder bejárás** esetében azok után írjuk ki.

Tétel: Ha x egy n csúcsú részfa gyökere, akkor az INORDER-FABEJÁRÁS(x) hívás $\Theta(n)$ ideig tart.

Hatékonyság: Egy n csúcsú bináris keresőfa bejárása $\Theta(n)$ ideig tart.

Feladatok

- Rajzoljunk 2, 3, 4, 5 és 6 magasságú bináris keresőfákat, amelyek az alábbi halmazban lévő kulcsokat tartalmazzák.
 - {1, 4, 5, 10, 16, 17, 21}
- Milyen kulcssorozatot írna ki az alábbi bináris keresőfa esetén egy preorder bejárást követő algoritmus, és milyen egy posztorder bejárást követő?



Bináris keresőfák

FÁBAN-KERES(x, k)

```

1  if  $x = \text{NIL}$  vagy  $k = \text{kulcs}[x]$ 
2      return  $x$ 
3  if  $k < \text{kulcs}[x]$ 
4      return FÁBAN-KERES( $\text{bal}[x], k$ )
5  else
6      return FÁBAN-KERES( $\text{jobb}[x], k$ )
    
```

FÁBAN-ITERATÍVAN-KERES(x, k)

```

1  while  $x \neq \text{NIL}$  és  $k \neq \text{kulcs}[x]$ 
2      if  $k < \text{kulcs}[x]$ 
3           $x \leftarrow \text{bal}[x]$ 
4      else
5           $x \leftarrow \text{jobb}[x]$ 
6  return  $x$ 
    
```

Hatékonyság: Egy h magasságú bináris keresőfában a keresés $O(h)$ ideig tart.

Feladatok

- Tegyük fel, hogy egy bináris keresőfában a kulcsok az 1 és 1000 közötti egész számok. A 363 értéket akarjuk megkeresni. Az alábbi sorozatok közül melyek nem lehetnek keresési sorozatok?
 - 2, 252, 401, 398, 330, 344, 397, 363
 - 924, 220, 911, 244, 898, 258, 362, 363
 - 925, 202, 911, 240, 912, 245, 363



Bináris keresőfák

FÁBAN-MINIMUM(x)

```
1  while  $bal[x] \neq \text{NIL}$ 
2       $x \leftarrow bal[x]$ 
3  return  $x$ 
```

FÁBAN-MAXIMUM(x)

```
1  while  $jobb[x] \neq \text{NIL}$ 
2       $x \leftarrow jobb[x]$ 
3  return  $x$ 
```

FÁBAN-KÖVETKEZŐ(x)

```
1  if  $jobb[x] \neq \text{NIL}$ 
2      return FÁBAN-MINIMUM( $jobb[x]$ )
3   $y \leftarrow szülő[x]$ 
4  while  $y \neq \text{NIL}$  és  $x = jobb[y]$ 
5       $x \leftarrow y$ 
6       $y \leftarrow szülő[y]$ 
7  return  $y$ 
```

Hatékonyság: Egy h magasságú bináris keresőfában ezek a műveletek is elvégezhetők $O(h)$ idő alatt.



Bináris keresőfák

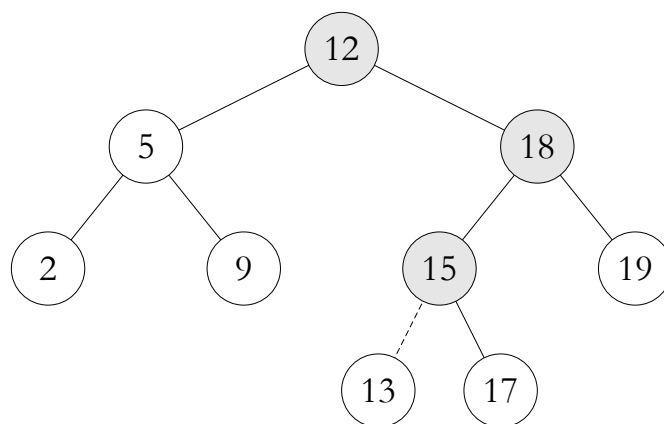
FÁBA-BESZÚR(T, z)

```

1   $y \leftarrow \text{NIL}$ 
2   $x \leftarrow \text{gyökér}[T]$ 
3  while  $x \neq \text{NIL}$ 
4       $y \leftarrow x$ 
5      if  $\text{kulcs}[z] < \text{kulcs}[x]$ 
6           $x \leftarrow \text{bal}[x]$ 
7      else
8           $x \leftarrow \text{jobb}[x]$ 
9   $\text{szülő}[z] \leftarrow y$ 
10 if  $y = \text{NIL}$ 
11      $\text{gyökér}[T] \leftarrow z$ 
12 else
13     if  $\text{kulcs}[z] < \text{kulcs}[y]$ 
14          $\text{bal}[y] \leftarrow z$ 
15     else
16          $\text{jobb}[y] \leftarrow z$ 
    
```

Hatékonyság: Egy h magasságú bináris keresőfában ez a művelet is $O(h)$ idejű.

Bináris keresőfák



Beszúrás bináris keresőfába

Feladatok

- Milyen magas bináris keresőfa épül fel, ha egy kezdetben üres fába rendre az alábbi kulcsokat szúrjuk be! Hányadik szintre kerül az utolsó kulcs, ha a gyökér szintje az első szint?
 - 2, 4, 5, 7, 1, 6, 3
- Adjuk meg az előbbi kulcsok egy olyan sorrendjét, amely minimális magasságú fát eredményez!
- Milyen a FÁBA-BESZÚR eljárás aszimptotikus viselkedése, ha a kezdetben üres fába n darab egyforma kulcsot szúrunk be?



Bináris keresőfák

FÁBÓL-TÖRÖL(T, z)

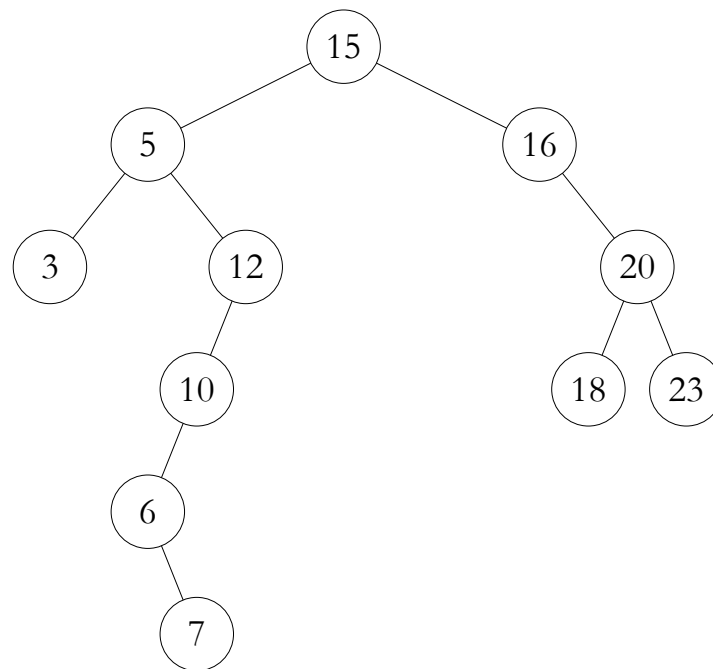
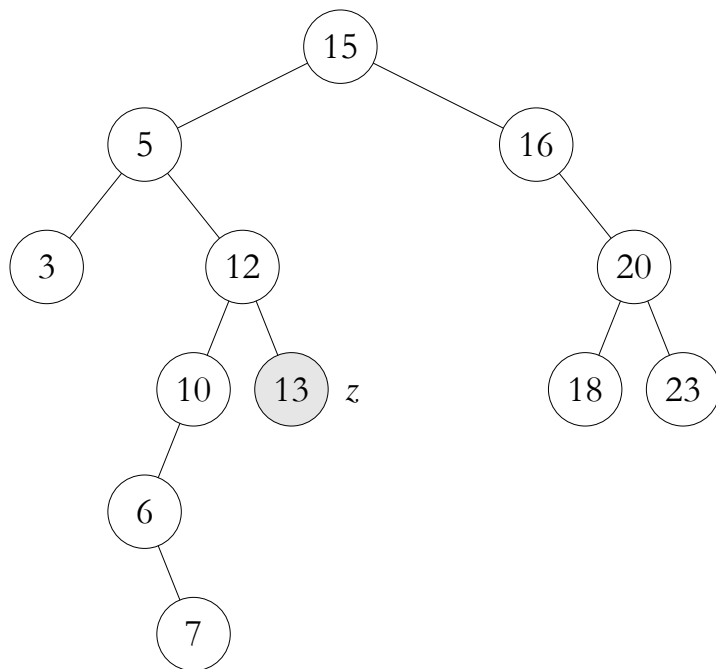
```

1  if  $bal[z] = \text{NIL}$  vagy  $jobb[z] = \text{NIL}$ 
2       $y \leftarrow z$ 
3  else
4       $y \leftarrow \text{FÁBAN-KÖVETKEZŐ}(z)$ 
5  if  $bal[y] \neq \text{NIL}$ 
6       $x \leftarrow bal[y]$ 
7  else
8       $x \leftarrow jobb[y]$ 
9  if  $x \neq \text{NIL}$ 
10      $szülő[x] \leftarrow szülő[y]$ 
11 if  $szülő[y] = \text{NIL}$ 
12      $gyökér[T] \leftarrow x$ 
13 else
14     if  $y = bal[szülő[y]]$ 
15          $bal[szülő[y]] \leftarrow x$ 
16     else
17          $jobb[szülő[y]] \leftarrow x$ 
18 if  $y \neq z$ 
19      $kulcs[z] \leftarrow kulcs[y]$ 
20     /*  $y$  kísérő adatait is másoljuk */
21 return  $y$ 
    
```

Hatékonyság: Egy h magasságú bináris keresőfában ez a művelet is $O(h)$ idejű.

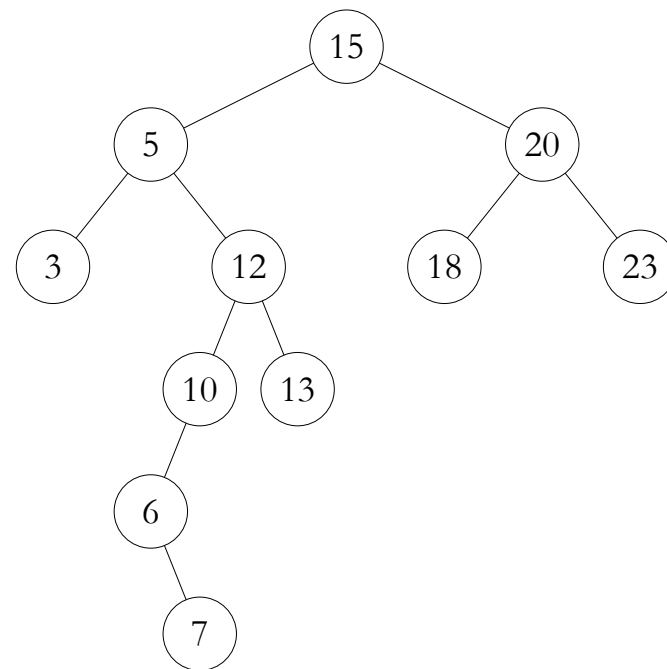
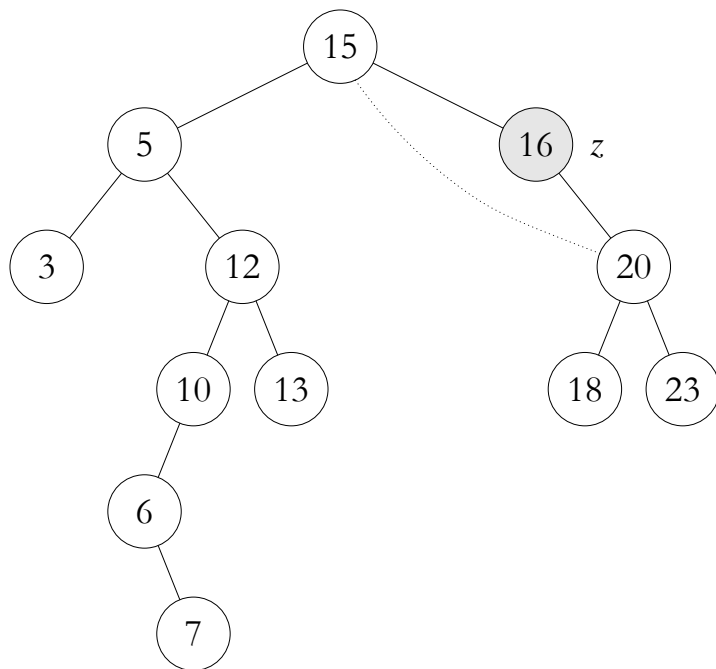


Bináris keresőfák



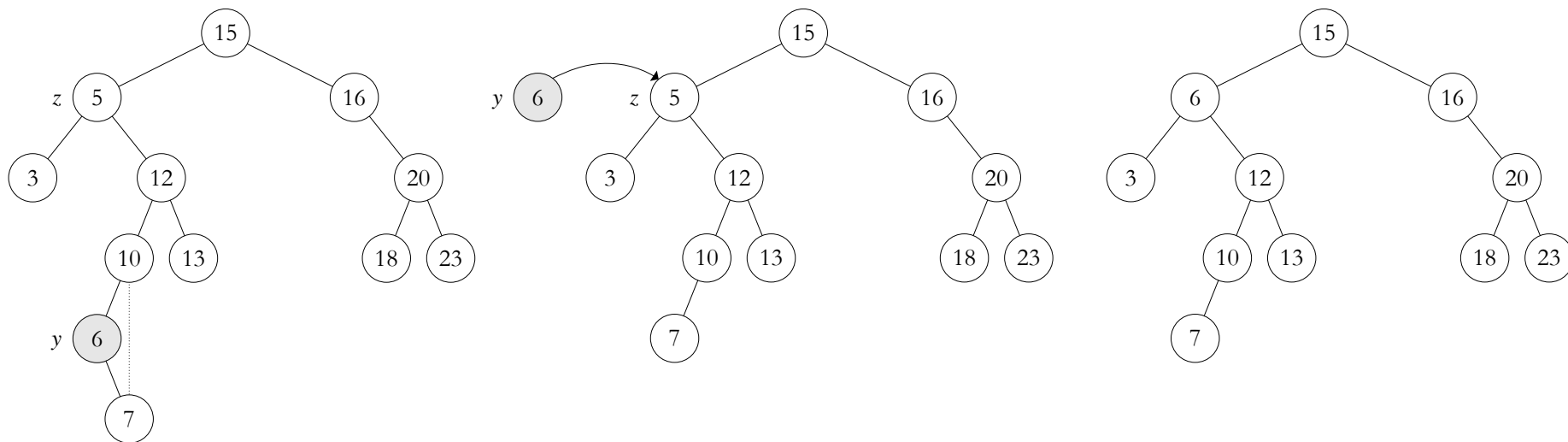
Törlés bináris keresőfából

Bináris keresőfák



Törlés bináris keresőfából

Bináris keresőfák



Törlés bináris keresőfából

Bináris keresőfák

Ha egy bináris keresőfa beszúrások és törlések sorozatával épül fel, akkor a fa átlagos magasságáról keveset mondhatunk, ellenben ha csak beszúrásokkal épül, akkor már többet.

Legyen adva n egymástól különböző kulcs, amelyekből bináris keresőfát építünk úgy, hogy a kulcsokat valamilyen sorrendben egymás után beszúrjuk a kezdetben üres fába.

Ha itt minden sorrend, vagyis az n kulcsnak mind az $n!$ permutációja egyformán valószínű, akkor a kapott fát **véletlen építésű bináris keresőfának** nevezzük.

Tétel: Egy n különböző kulcsot tartalmazó véletlen építésű bináris keresőfa várható magassága $O(\lg n)$.

Feladatok

- Milyen bináris keresőfát kapunk, ha egy kezdetben üres fába rendre beszúrjuk a 4, 10, 7, 9, 15, 2, 5, 8, 6, 1, 3 kulcsokat, majd rendre töröljük az 1, 5, 7, és 4 kulcsokat? Adjuk meg a beszúrásuk után előállt fát és az egyes törlések után előállt fákat!
- Igaz-e, hogy ha egy bináris keresőfa valamely csúcsának van két gyereke, akkor a rákövetkezőjének nincs baloldali gyereke és a megelőzőjének nincs jobboldali gyereke?
- Egy halmaz elemeit rendezhetjük úgy is, hogy először építünk egy, a halmaz elemeit tartalmazó bináris keresőfát (az elemeket egymás után beszúrjuk egy kezdetben üres fába), majd a fa tartalmát az inorder bejárással kiírjuk. Mi ennek a rendezési eljárásnak a legrosszabb és a legjobb futási ideje?
- Igaz-e, hogy a törlés kommutatív abban az értelemben, hogy egy bináris keresőfából előbb x -et, majd y -t törölve ugyanahhoz a fához jutunk, mint ha előbb töröljük a fából y -t, és azután az x -et?



Piros-fekete fák

A **piros-fekete fa** olyan bináris keresőfa, amelynek minden csúcsa egy extra bit információt tartalmaz, ez a csúcs színe, amelynek értékei: PIROS, FEKETE.

A csúcsok színezésének korlátozásával biztosítható, hogy a piros-fekete fában bármely, a gyökértől levélig vezető út hossza nem lehet nagyobb, mint a legrövidebb ilyen út hosszának kétszerese.

Az ilyen fák megközelítőleg **kiegyensúlyozottak**, lehetővé teszik, hogy az alapvető dinamikus halmaz műveletek a legrosszabb esetben is $O(\lg n)$ idejűek legyenek.

Egy bináris keresőfa piros-fekete fa, ha rendelkezik a következő **piros-fekete tulajdonságokkal**:

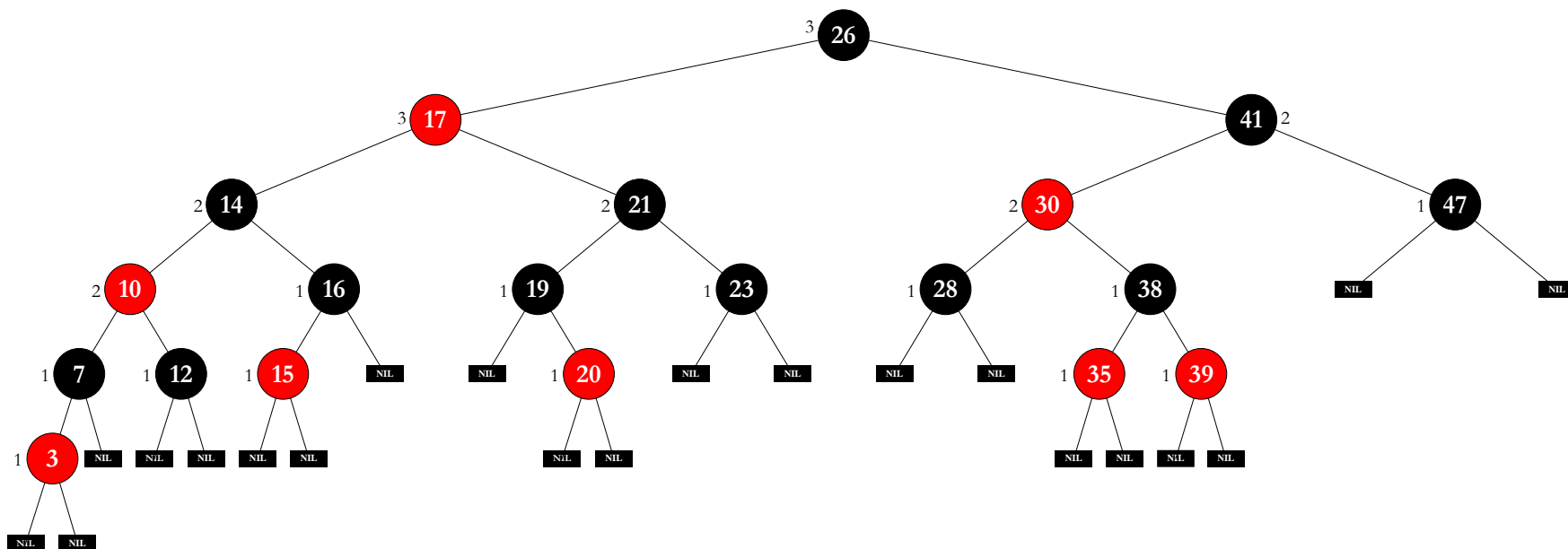
1. Minden csúcs színe vagy piros, vagy fekete.
2. A gyökércsúcs színe fekete.
3. Minden levél (NIL) színe fekete.
4. Minden piros csúcsnak mindkét gyereke fekete.
5. Minden csúcsra igaz, hogy a belőle kiinduló, levélig vezető utakon ugyanannyi fekete csúcs van.

Egy x csúcs **fekete-magasságának** nevezzük az x csúcsból induló, levélig vezető úton található, x -et nem tartalmazó fekete csúcsok számát.

Egy piros-fekete fa fekete-magassága a fa gyökércsúcsának fekete-magassága.

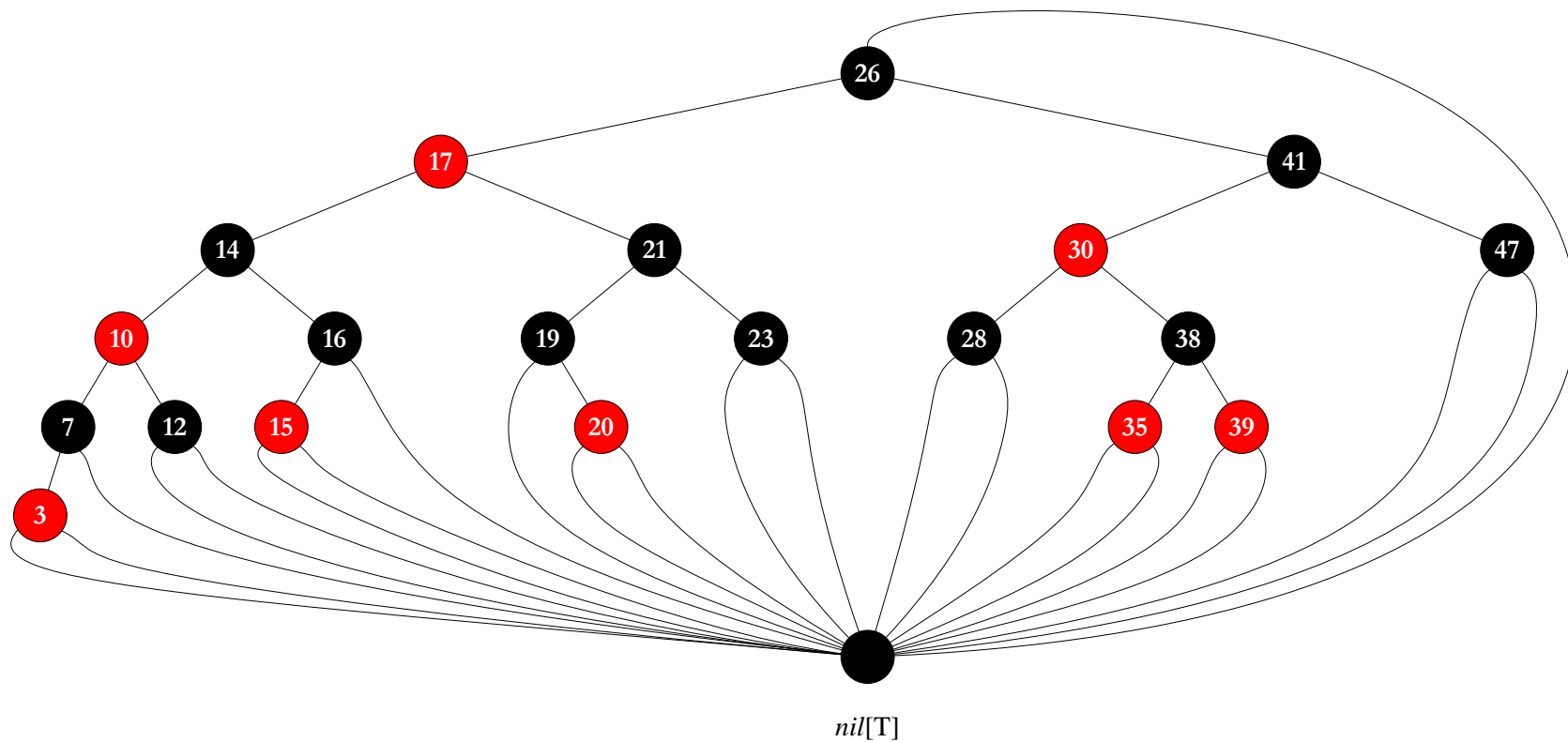
Tétel: Bármely n belső csúcsot tartalmazó piros-fekete fa magassága legfeljebb $2\lg(n+1)$.

Piros-fekete fák



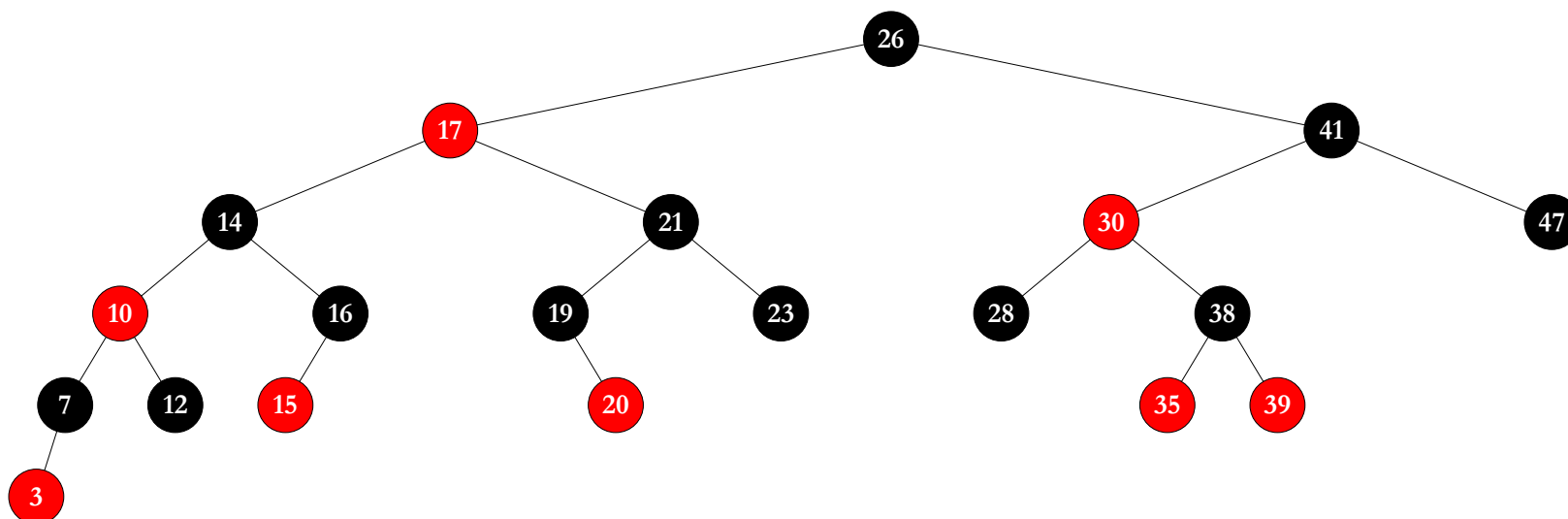
Egy piros-fekete fa a fekete magasságokkal

Piros-fekete fák



Egy piros-fekete fa egy $nil[T]$ őrszemmel

Piros-fekete fák

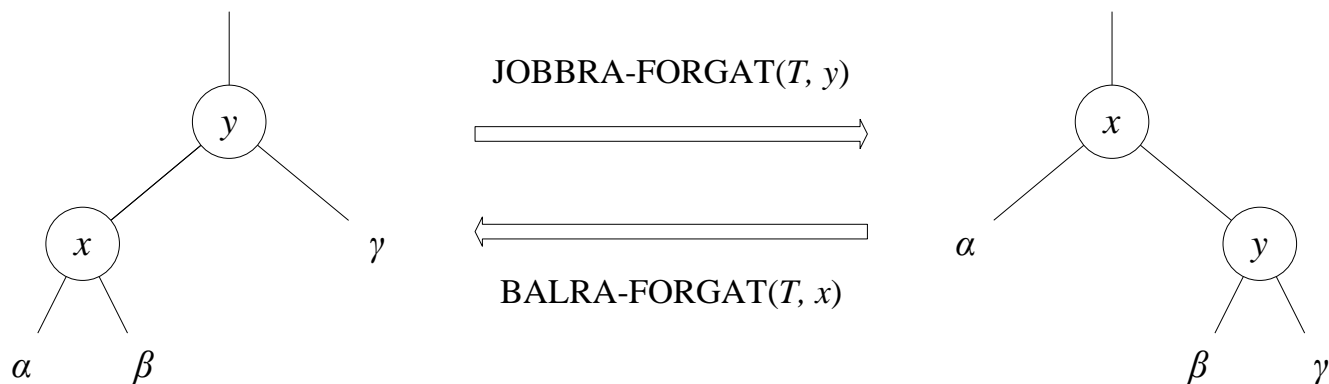


Egy piros-fekete fa a levelek és a gyökér szülője nélkül

Feladatok

- Rajzoljuk meg az $\{1, \dots, 15\}$ kulcsokat tartalmazó 3 magasságú teljes bináris keresőfát! Egészítsük ki NIL levelekkel és színezzük be a csúcsokat úgy, hogy a fa fekete-magassága 2, 3, ill. 4 legyen!
- Rajzoljuk le azt a bináris fát, amely az előző dián lévő piros-fekete fából keletkezik, ha a fát a FÁBA-BESZÚR eljárás a 36 kulccsal bővíti! Ha az új csúcs színét pirosra állítjuk, piros-fekete fát kapunk-e? Mi a helyzet, ha az új csúcsot feketére színezzük?
- Hogyan adjunk meg egy n csúcsú piros-fekete fát, hogy a lehető legnagyobb legyen a piros és a fekete belső csúcsok aránya? Milyen fa esetén lesz az arány a legkisebb, és mi lesz ez az érték?

Piros-fekete fák



Forgató műveletek piros-fekete fákon

Piros-fekete fák

BALRA-FORGAT(T, x)

```

1   $y \leftarrow jobb[x]$ 
2   $jobb[x] \leftarrow bal[y]$ 
3  if  $bal[y] \neq nil[T]$ 
4       $szülő[bal[y]] \leftarrow x$ 
5   $szülő[y] \leftarrow szülő[x]$ 
6  if  $szülő[x] = nil[T]$ 
7       $gyökér[T] \leftarrow y$ 
8  else
9      if  $x = bal[szülő[x]]$ 
10          $bal[szülő[x]] \leftarrow y$ 
11      else
12          $jobb[szülő[x]] \leftarrow y$ 
13   $bal[y] \leftarrow x$ 
14   $szülő[x] \leftarrow y$ 
    
```

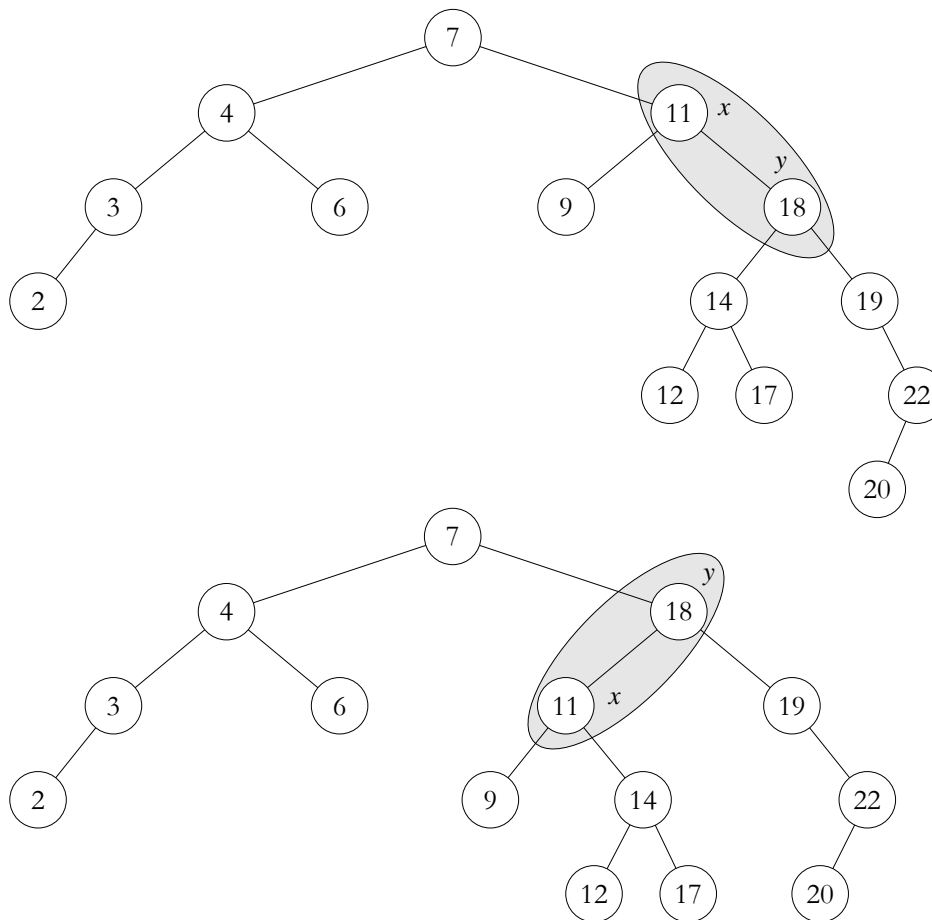
Megjegyzés:

- Az eljárás feltételezi, hogy $jobb[x] \neq nil[T]$, és a gyökér szülője $nil[T]$.
- A JOBBRA-FORGAT művelet analóg módon megvalósítható.

Hatékonyság: Mindkét forgatási művelet $O(1)$ időigényű.



Piros-fekete fák



A BALRA-FORGAT(T, x) működése

Feladatok

- Rajzoljuk le azt a bináris keresőfát, amelyet az előző dia alsó bináris keresőfájából kapunk akkor, ha a gyökércsúcsnál balra forgatást végzünk!
- Milyen fát kapunk akkor, ha a gyökércsúcsnál jobbra forgatást végzünk!



Bináris keresőfák

PF-FÁBA-BESZÚR(T, z)

```

1   $y \leftarrow nil[T]$ 
2   $x \leftarrow gyökér[T]$ 
3  while  $x \neq nil[T]$ 
4       $y \leftarrow x$ 
5      if  $kulcs[z] < kulcs[x]$ 
6           $x \leftarrow bal[x]$ 
7      else
8           $x \leftarrow jobb[x]$ 
9   $szülő[z] \leftarrow y$ 
10 if  $y = nil[T]$ 
11      $gyökér[T] \leftarrow z$ 
12 else
13     if  $kulcs[z] < kulcs[y]$ 
14          $bal[y] \leftarrow z$ 
15     else
16          $jobb[y] \leftarrow z$ 
17  $bal[z] \leftarrow nil[T]$ 
18  $jobb[z] \leftarrow nil[T]$ 
19  $szín[z] \leftarrow PIROS$ 
20 PF-FÁBA-BESZÚR-JAVÍT( $T, z$ )
    
```



Piros-fekete fák

PF-FÁBA-BESZÚR-JAVÍT(T, z)

```

1  while  $szín[szülő[z]] = \text{PIROS}$ 
2      if  $szülő[z] = bal[szülő[szülő[z]]]$ 
3           $y \leftarrow jobb[szülő[szülő[z]]]$ 
4          if  $szín[y] = \text{PIROS}$ 
5               $szín[szülő[z]] \leftarrow \text{FEKETE}$  /* 1. eset */
6               $szín[y] \leftarrow \text{FEKETE}$  /* 1. eset */
7               $szín[szülő[szülő[z]]] \leftarrow \text{PIROS}$  /* 1. eset */
8               $z \leftarrow szülő[szülő[z]]$  /* 1. eset */
9      else
10         if  $z = jobb[szülő[z]]$ 
11              $z \leftarrow szülő[z]$  /* 2. eset */
12             BALRA-FORGAT( $T, z$ ) /* 2. eset */
13              $szín[szülő[z]] \leftarrow \text{FEKETE}$  /* 3. eset */
14              $szín[szülő[szülő[z]]] \leftarrow \text{PIROS}$  /* 3. eset */
15             JOBBRA-FORGAT( $T, szülő[szülő[z]]$ ) /* 3. eset */
16     else
17         ugyanaz, mint az igaz ág, csak a „bal” és a „jobb” felcserélve
18      $szín[gyökér[T]] \leftarrow \text{FEKETE}$ 

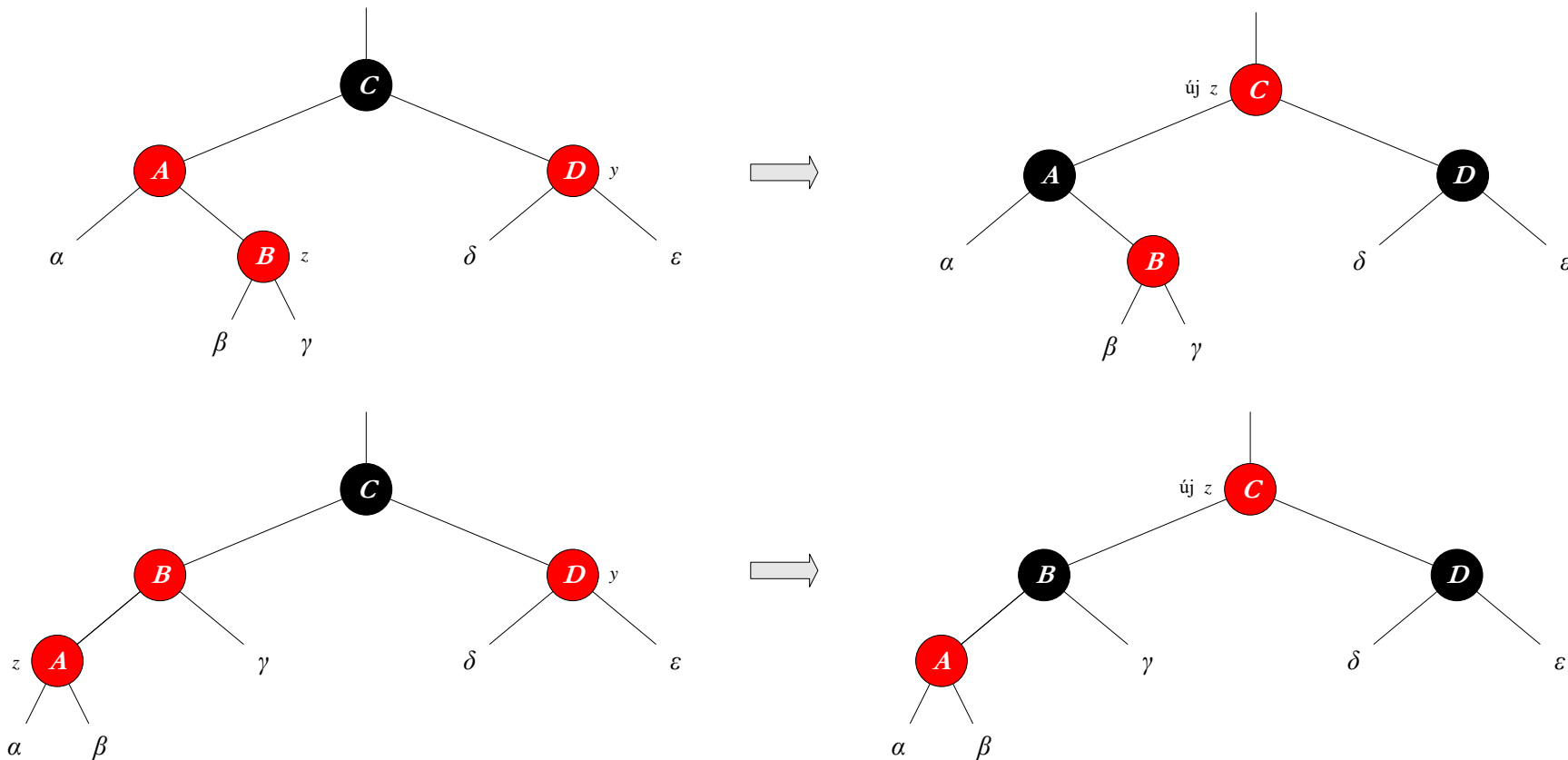
```

Megjegyzés: Vegyük észre, hogy ha $szín[szülő[z]] = \text{PIROS}$, akkor létezik a $szülő[szülő[z]]$ csúcs.

Hatékonyság: A művelet elvégezhető $O(\lg n)$ időben, ha a fa n csúcsot tartalmaz.

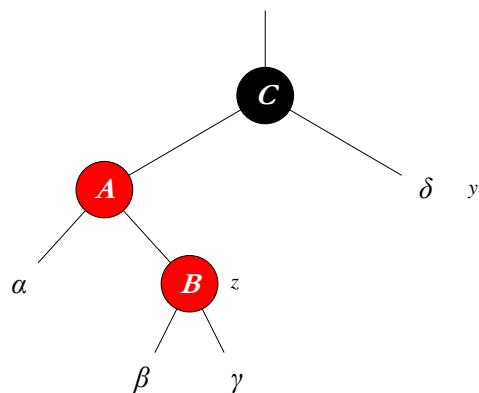


Piros-fekete fák

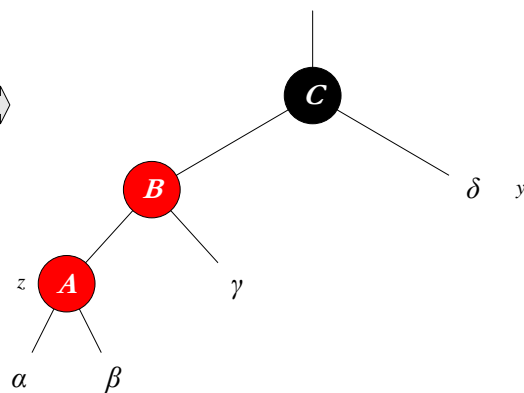


A PF-FÁBA-BESZÚR-JAVÍT eljárás 1. esete

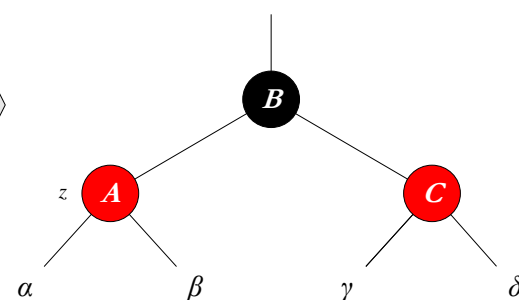
Piros-fekete fák



2. eset

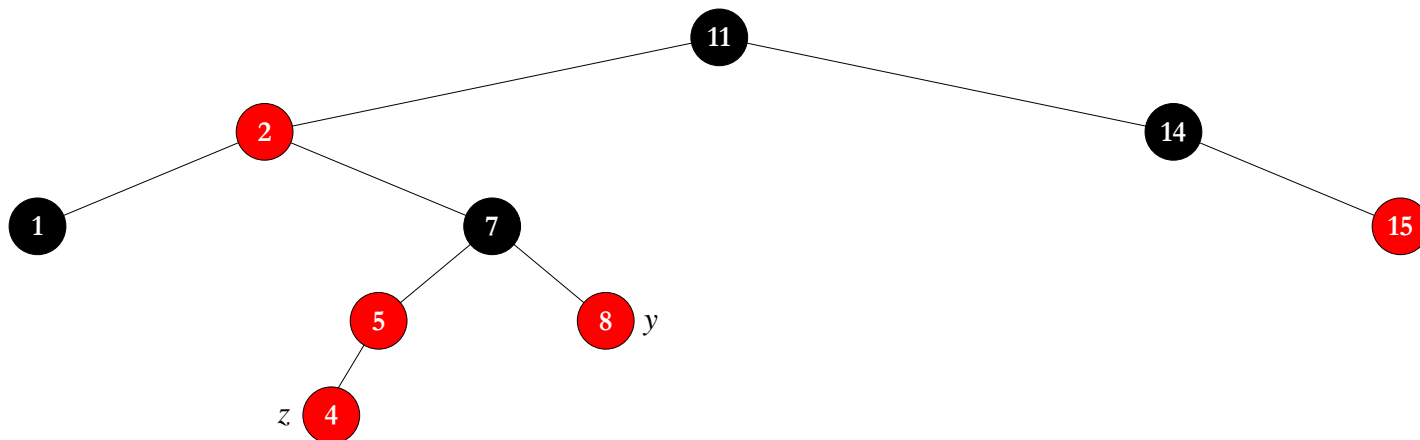


3. eset

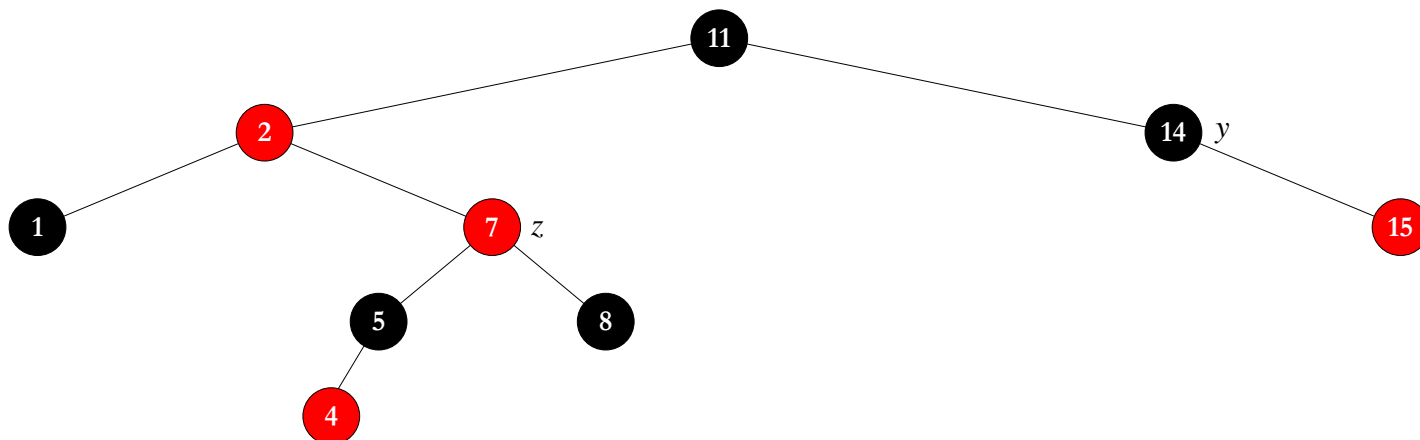


A PF-FÁBA-BESZÚR-JAVÍT eljárás 2. és 3. esete

Piros-fekete fák



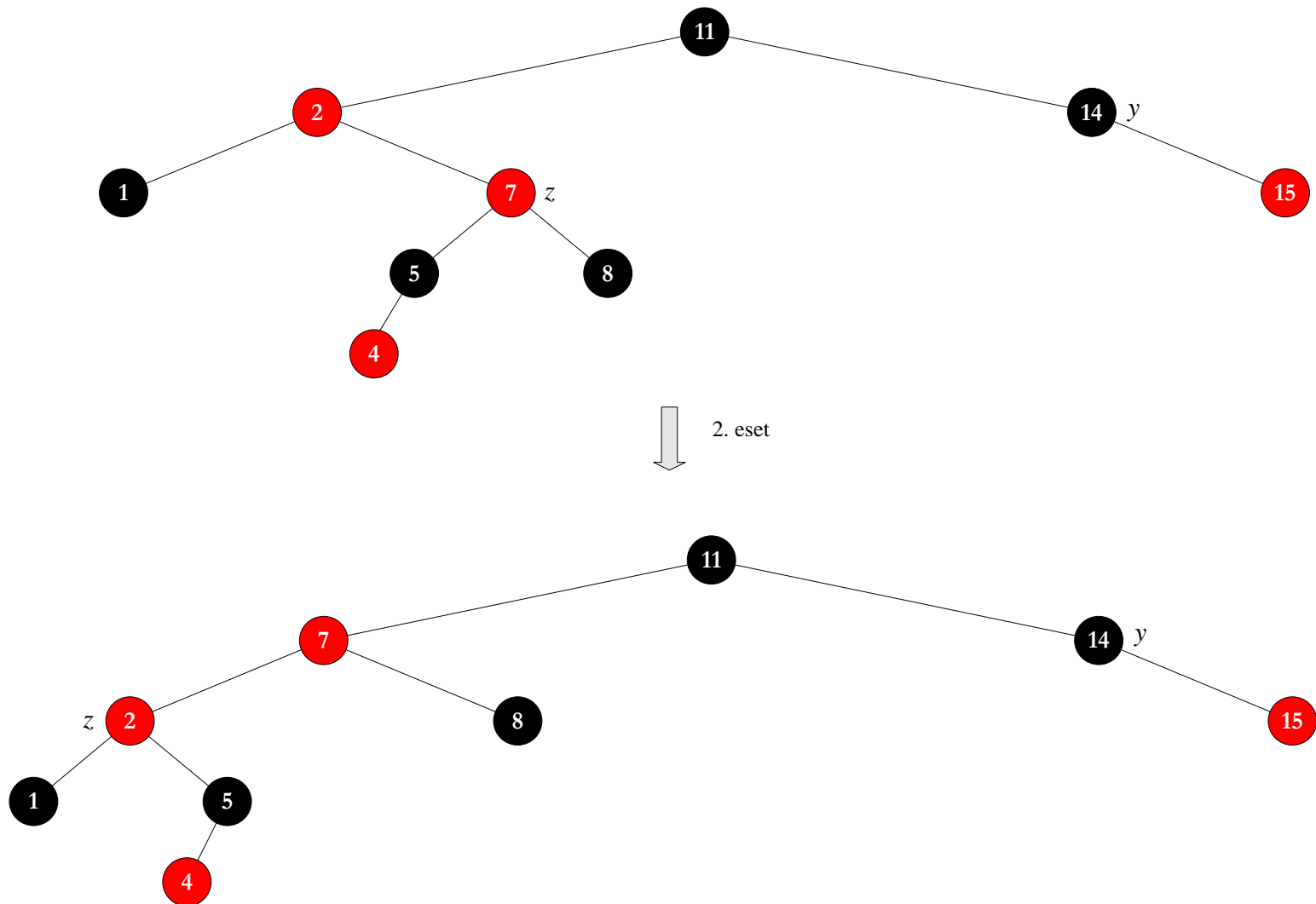
1. eset



A PF-FÁBA-BESZÚR-JAVÍT működése

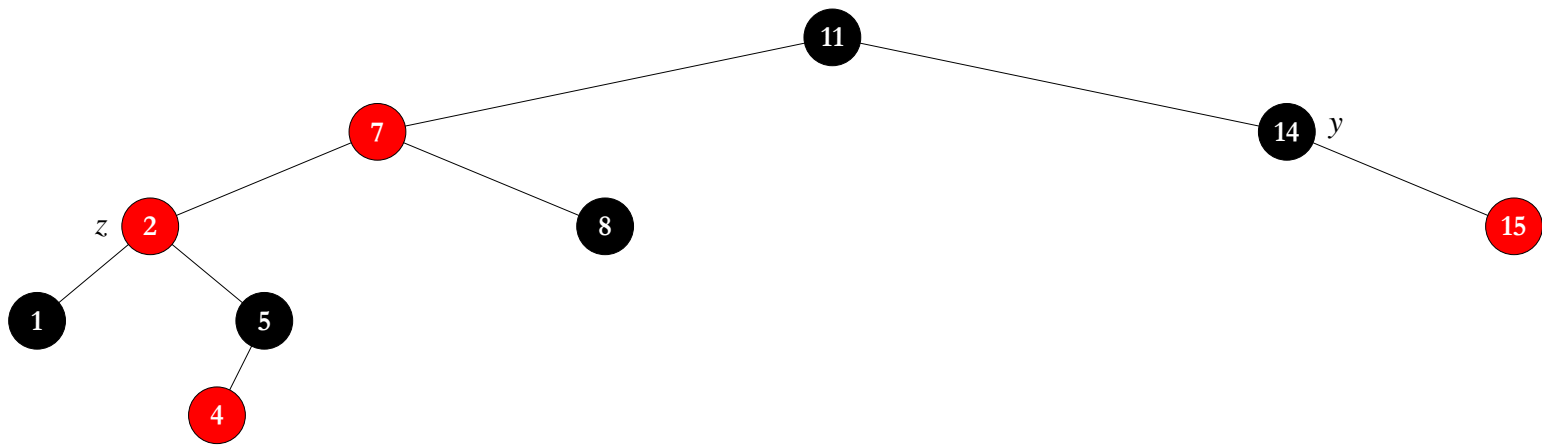


Piros-fekete fák

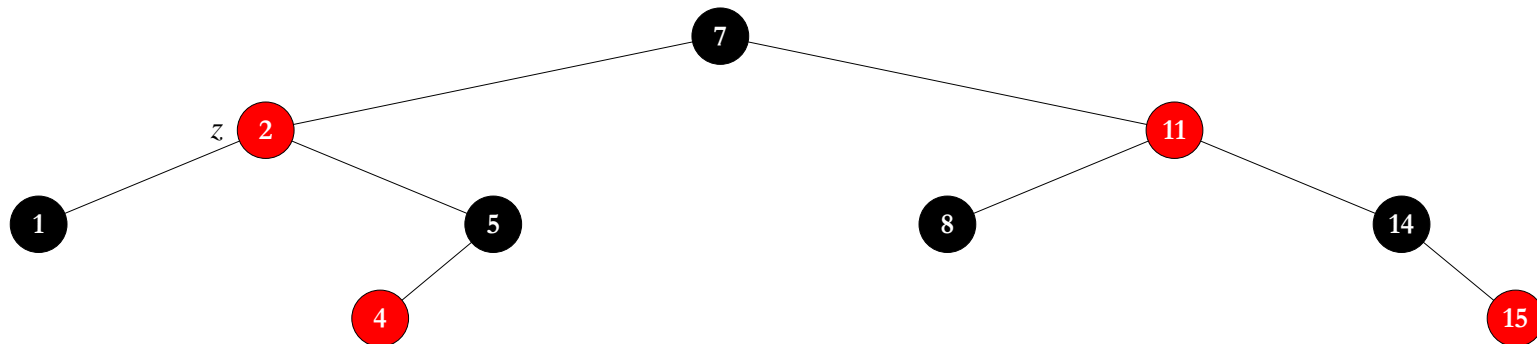


A PF-FÁBA-BESZÚR-JAVÍT működése

Piros-fekete fák



3. eset



A PF-FÁBA-BESZÚR-JAVÍT működése



Feladatok

- A PF-FÁBA-BESZÚR eljárásban az új z csúcs színét pirosra állítjuk. Vegyük észre, hogy ha z színét feketére állítanánk, akkor teljesülne a piros-fekete fák 4. tulajdonsága. Miért nem fekete színt kap az új csúcs?
- Rajzoljuk le azt a bináris keresőfát, amelyet az előző dia legalsó piros-fekete fájából kapunk akkor, ha a 3 kulcsú elemet a fába szúrjuk! Jelezzük a csúcsok színét P ill. F betűkkel a csúcsok mellett! Tüntessük fel a csúcsok mellett a csúcsok fekete-magasságát is!
- Rajzoljuk le azt a piros-fekete fát, amely a kezdetben üres fából úgy keletkezik, hogy egymás után bővítjük a fát a 15, 20, 25, 18, 12, 6, 8, 3, 4 kulcsokkal! Jelezzük a csúcsok színét P ill. F betűkkel a csúcsok mellett! Mekkora a fa fekete-magassága?

Piros-fekete fák

Megjegyzés:

- A piros-fekete fákból való törlés (PF-FÁBÓL-TÖRÖL) lényegében megegyezik a bináris keresőfákból való törléssel (FÁBÓL-TÖRÖL), de ha fekete csúcsot kapcsolunk ki a fából, akkor a megsérülő piros-fekete tulajdonságokat helyre kell állítanunk.
- A helyreállító eljárás (PF-FÁBÓL-TÖRÖL-JAVÍT) időigénye $O(\lg n)$ és legfeljebb három forgatást végez.

Következmény: A piros-fekete fákból való törlés is $O(\lg n)$ időigényű.

Adatszerkezetek kibővítése

- Algoritmusok tervezése során gyakran előfordul egy adott **adatszerkezet bővítése**. Ezt a bővítendő adatszerkezetet **alap-adatszerkezetnek** nevezzük.

Egy adatszerkezet bővítésének **lépései**:

1. Az alap adatszerkezet megválasztása.
2. Az alap-adatszerkezetben fenntartandó kiegészítő információk meghatározása.
3. Annak igazolása, hogy a kiegészítő információk fenntarthatók az alap-adatszerkezet módosító műveletei során.
4. Új műveletek kifejlesztése.

Példák:

- A piros-fekete fák bővítése úgy, hogy hatékonyan támogassák a dinamikus halmazok rendezett mintára vonatkozó műveleteit.
- A piros-fekete fák bővítése intervallumokat tartalmazó dinamikus halmazok kezelésére.

Tétel: Legyen f piros-fekete fák olyan mezője, amely bővítése az adatszerkezetnek és teljesül, hogy $f[x]$ értéke kiszámítható az x , $bal[x]$ és $jobb[x]$ pontokban lévő információkból, beleértve az $f[bal[x]]$ és $f[jobb[x]]$ értékeket. Ekkor f értéke fenntartható a beszúrás és törlés során úgy, hogy ezen műveletek aszimptotikus hatékonysága nem változik, azaz $O(\lg n)$ minden n csúcsú fára.

Adatszerkezetek kibővítése

- Egy n elemű halmaz esetén a **rendezett minta** i -edik ($1 \leq i \leq n$) eleme a halmaz i -edik legkisebb eleme.
- Egy nem rendezett halmaz i -edik legkisebb eleme megkereshető $O(n)$ idejű algoritmussal.
- A piros-fekete fák kibővítésével ez a feladat megoldható $O(\lg n)$ időben.
- Egy adott elem **rangja** (a rendezett halmazbeli pozíciója) szintén meghatározható lesz $O(\lg n)$ időben.

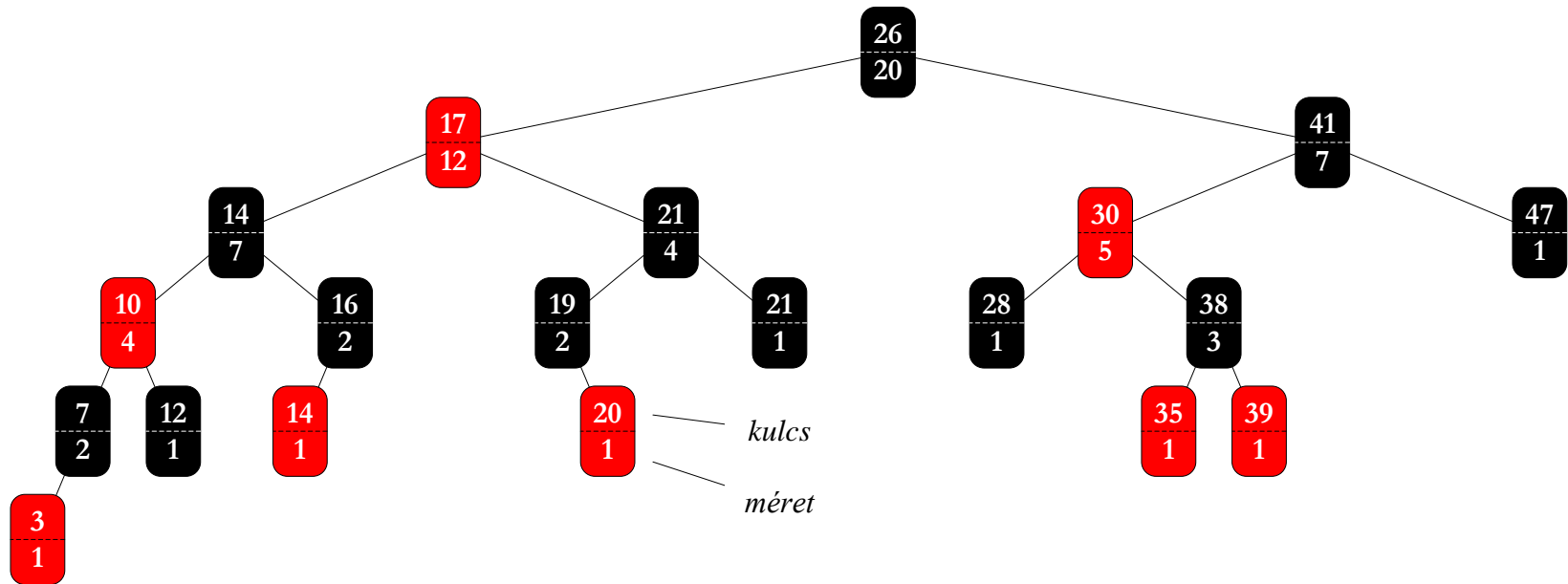
A **rendezettminta-fa** olyan piros-fekete fa, amelynek minden x pontjában a szokásos mezőkön ($kulcs[x]$, $bal[x]$, $jobb[x]$, $szülő[x]$) kívül egy kiegészítő információt tároló mező, a $méret[x]$ is szerepel.

Ez a mező az x gyökerű részfa (belső) pontjainak számát tartalmazza (beleértve saját magát is), tehát a részfa méretét.

Ha a $méret[NIL] = 0$ definíciót használjuk, akkor $méret[x] = méret[bal[x]] + méret[jobb[x]] + 1$.

Megjegyzés: Azonos kulcsok lehetnek, ekkor egy elem rangja legyen az inorder bejárás szerinti sorszáma (pl. a következő ábrán a fekete csúcsban lévő 14 kulcsú elem rangja 5, míg a piros csúcsbelié 6).

Adatszerkezetek kibővítése



Egy rendezettminta-fa

Adatszerkezetek kibővítése

RM-KIVÁLASZT(x, i)

```

1   $r \leftarrow \text{méret}[\text{bal}[x]] + 1$ 
2  if  $i = r$ 
3      return  $x$ 
4  else
5      if  $i < r$ 
6          return RM-KIVÁLASZT( $\text{bal}[x], i$ )
7  else
8      return RM-KIVÁLASZT( $\text{jobb}[x], i - r$ )
    
```

Megjegyzés: Egy T rendezettminta-fa i -edik legkisebb elemét az RM-KIVÁLASZT($\text{gyökér}[T], i$) hívás adja.

Hatékonyság: Az eljárás $O(\lg n)$ időigényű, minden n elemű dinamikus halmazra.

Feladatok

- Melyik csúcsot adja eredményül a példabeli T rendezettminta-fa esetén az $\text{RM-KIVÁLASZT}(\text{gyökér}[T], 7)$ hívás?
- Milyen csúcsokat érintünk az $\text{RM-KIVÁLASZT}(\text{gyökér}[T], 16)$ hívás hatására ugyanebben a T fában?

Adatszerkezetek kibővítése

RM-RANG(T, x)

```

1   $r \leftarrow \text{méret}[\text{bal}[x]] + 1$ 
2   $y \leftarrow x$ 
3  while  $y \neq \text{gyökér}[T]$ 
4      if  $y = \text{jobb}[\text{szülő}[y]]$ 
5           $r \leftarrow r + \text{méret}[\text{bal}[\text{szülő}[y]]] + 1$ 
6       $y \leftarrow \text{szülő}[y]$ 
7  return  $r$ 
    
```

Az RM-RANG algoritmus a T rendezettminta-fa x csúcsára meghatározza a csúcs pozícióját a fa inorder bejárásával kapott sorozatban.

Hatékonyság: Az eljárás $O(\lg n)$ időigényű, minden n elemű dinamikus halmazra.

Feladatok

- Milyen eredményt ad a példabeli T rendezettminta-fa és a 19-es kulcsú csúcs esetén az RM-RANG függvény?
- Milyen csúcsokat érintünk az RM-RANG függvényben ugyanezen T fa és a 35-ös kulcsú csúcs esetén?



Adatszerkezetek kibővítése

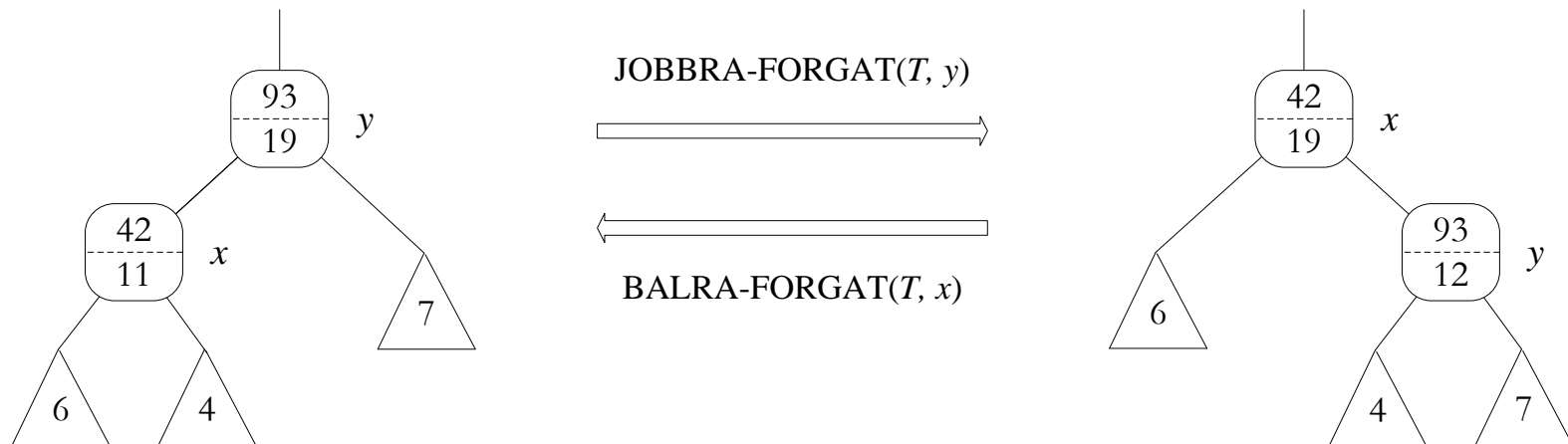
A részfák méret tulajdonsága fenntartható a beszúrás és törlés során úgy, hogy a műveletek aszimptotikus időigénye nem változik.

A BALRA-FORGAT eljárás végére beszúrando az alábbi két sor:

15 $méret[y] \leftarrow méret[x]$

16 $méret[x] \leftarrow méret[bal[x]] + méret[jobb[x]] + 1$

A JOBBRA-FORGAT eljárás módosítása a szimetriának megfelelően történik.



A méret információ fenntartása

Adatszerkezetek kibővítése

Feladat: Piros-fekete fák kibővítésével készítsünk intervallumokat tartalmazó adatstruktúrát, amelyek támogatják a dinamikus halmaz műveleteket!

Alkalmazás: Intervallumokkal kényelmesen ábrázolhatók események, amelyek időben egy folytonos szakaszt képeznek, így pl. kérdéseket tehetünk fel:

- Az intervallum adatbázisban mely események történtek egy adott intervallumon belül?
- Van-e az adatbázisban egy adott intervallummal átfedő intervallum?

Zárt intervallumon egy $[t_1, t_2]$ rendezett valós számpárt értünk, ahol $t_1 \leq t_2$. A $[t_1, t_2]$ intervallum tehát a $\{t \in \mathbb{R}: t_1 \leq t \leq t_2\}$ halmazt ábrázolja.

Nyitott, ill. félig nyitott intervallum esetén mindkét, ill. valamelyik végpont nem szerepel a halmazban.

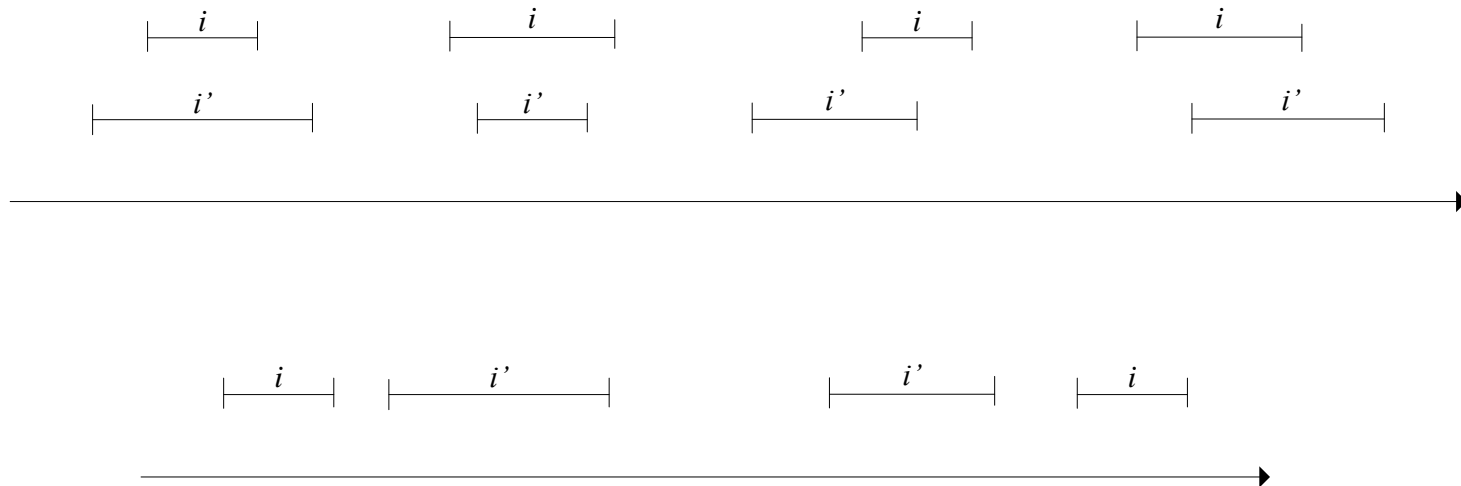
Egy $[t_1, t_2]$ intervallumot ábrázolhatunk egy olyan i objektummal, amelynek két mezője van, az intervallum két végpontja: $alsó[i] = t_1$ (**alsó végpont**) és $felső[i] = t_2$ (**felső végpont**).

Az i és i' intervallumok **átfedik egymást**, ha az $i \cap i'$ halmaz nem üres, vagyis $alsó[i] \leq felső[i']$ és $alsó[i'] \leq felső[i]$.

Adatszerkezetek kibővítése

Bármely két, i és i' intervallum teljesíti az **intervallum trichotómia** tulajdonságot, azaz a következő állítások közül pontosan egy teljesül rájuk:

- Az i és i' intervallumok átfedik egymást,
- $felső[i] < alsó[i']$,
- $felső[i'] < alsó[i]$.



Intervallum trichotómia

Adatszerkezetek kibővítése

Az **intervallum-fa** olyan piros-fekete fa, amely lehetővé teszi olyan dinamikus halmaz kezelését, amelynek minden x eleme tartalmaz egy $int[x]$ intervallumot.

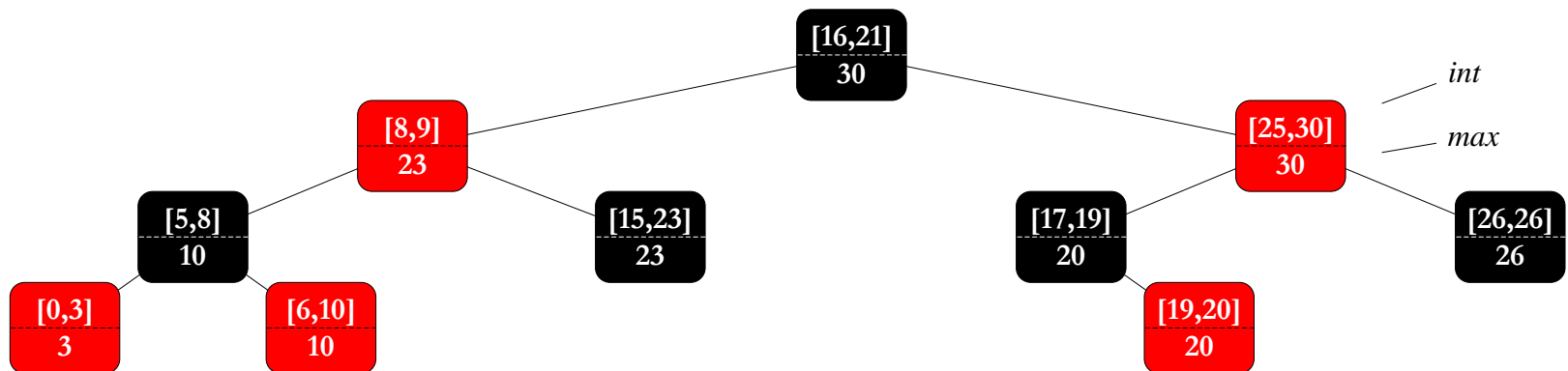
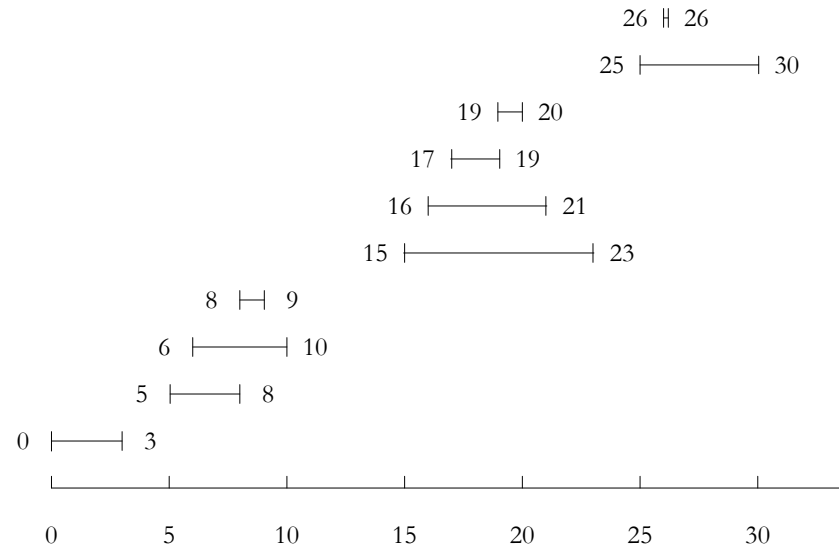
Az intervallum-fák a következő műveleteket támogatják:

- **INTERVALLUMOT-KERES(T, i)**: eredményül egy olyan x csúcsra mutató mutatót ad a T fában, amelyre teljesül, hogy $int[x]$ átfedi az i intervallumot, ill. $nil[T]$ az eredmény, ha nincs a fában ilyen pont.
- **INTERVALLUMOT-BESZÚR(T, x)**: bővíti a T intervallum-fát az x csúccsal, amelynek int mezője egy intervallumot tartalmaz.
- **INTERVALLUMOT-TÖRÖL(T, x)**: törli a T intervallum-fából az x csúcsot.

Az adatszerkezet bővítésének lépései:

1. Az alap-adatszerkezet: olyan intervallum-fa, amelyben minden x elem kulcsa az $alsó[int[x]]$.
2. Kiegészítő információ: $max[x]$, az x gyökerű részfában lévő intervallumok végpontjainak a maximuma.
3. Fenntarthatóság: mivel $max[x] = \max(felső[int[x]], max[bal[x]], max[jobb[x]])$, ezért (a korábbi tétel alapján) ez a tulajdonság fenntartható az alap-adatszerkezet módosító műveletei (beszúrás, törlés) során.
4. Az új művelet: **INTERVALLUMOT-KERES(T, i)**.

Adatszerkezetek kibővítése



Egy intervallum-fa

Adatszerkezetek kibővítése

INTERVALLUMOT-KERES(T, i)

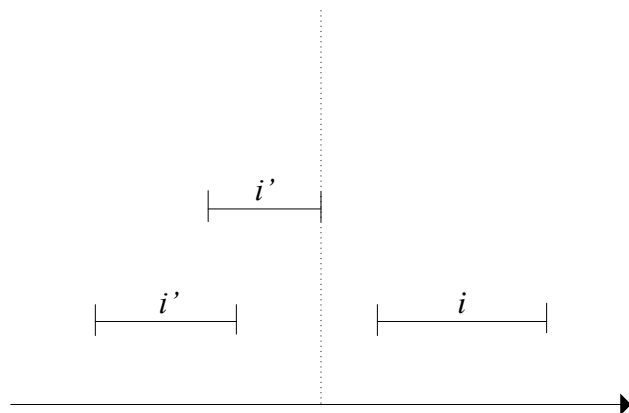
```

1   $x \leftarrow \text{gyökér}[T]$ 
2  while  $x \neq \text{nil}[T]$  és  $i$  nem fedi át  $\text{int}[x]$ -et
3      if  $\text{bal}[x] \neq \text{nil}[T]$  és  $\text{max}[\text{bal}[x]] \geq \text{alsó}[i]$ 
4           $x \leftarrow \text{bal}[x]$ 
5      else
6           $x \leftarrow \text{jobb}[x]$ 
7  return  $x$ 
    
```

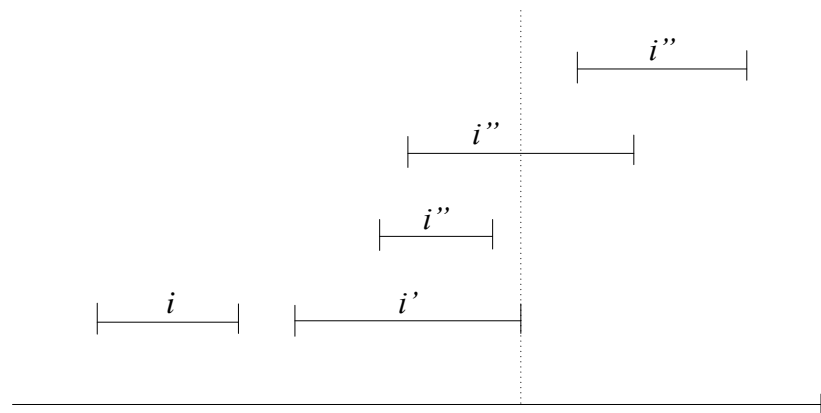
Hatékonyság: Az eljárás $O(\lg n)$ időigényű, minden n elemű dinamikus halmazra.

Megjegyzés: A keresésnél elegendő egyetlen, gyökértől induló utat vizsgálni.

Adatszerkezetek kibővítése



A bal ágon nem lehet, ezért
a jobb ágon keressük



Ha a bal ágon nem találjuk, akkor
a jobb ágon se lett volna

Az INTERVALLUMOT-KERES helyessége

Feladatok

- Milyen eredményt ad a példabeli T intervallum-fa és az $i=[4, 7]$ intervallum esetén az INTERVALLUMOT-KERES függvény?
- Milyen csúcsokat érintünk az INTERVALLUMOT-KERES függvényben ugyanezen T fa és az $i=[11, 14]$ intervallum esetén?
- Hogyan változna meg az INTERVALLUMOT-KERES függvény, ha a T fában lévő intervallumok nyitottak?
- Melyik átfedő intervallumot adja eredményül az INTERVALLUMOT-KERES függvény? Mi lenne, ha az átfedő intervallumok közül azt kellene eredményül adni, amelynek alsó végpontja a legkisebb?

B-fák

- A B-fák olyan kiegyensúlyozott keresőfák, amelyeket úgy terveztek, hogy hatékonyan lehessen alkalmazni őket mágneslemezeken (vagy más, közvetlen hozzáférésű másodlagos tároló berendezéseken).
- A B-fák hasonlóak a piros-fekete fákhhoz, de a lemezen kevesebb beviteli és kiviteli műveletet igényelnek.
- A B-fában a csúcsoknak sok gyerekük lehet, a gyerekek száma néhánytól több ezerig terjedhet, azaz a B-fában az „elágazási tényező” igen nagy lehet, amire a felhasznált mágneslemez jellemzői egy felső korlátot adnak.
- Egy n -csúcsú B-fa magassága $O(\lg n)$ így a B-fákkal megvalósított dinamikus halmazműveletek szintén $O(\lg n)$ idejűek.
- A B-fák tipikus alkalmazásaiban a kezelt adatmennyiség akkora, hogy rendszerint nem fér el egyszerre a főmemóriában. A B-fa algoritmusok csak azokat a blokkokat olvassák be a mágneslemezről a memóriába, amelyekre szükség van, és csak a megváltoztatott tartalmú blokkokat írják vissza a memóriából a mágneslemezre.

B-fák

Egy objektummal kapcsolatos művelet tipikus mintája a következő:

- 1 $x \leftarrow$ az objektum mutatója
- 2 LEMEZRŐL-OLVAS(x) /* Ha x már a memóriában van, akkor itt „no-operation” történik */
- 3 azok a műveletek, amelyek az x mezőit olvassák vagy módosítják
- 4 LEMEZRE-ÍR(x) /* Kimarad, ha az x egyik mezője sem változott meg */
- 5 további olyan műveletek, amelyek az x mezőit csak olvassák, de már nem módosítják

Megjegyzés:

- A B-fa egy csúcsának nagysága rendszerint a mágneslemez egy blokkjának a nagyságával egyezik meg, ezért a B-fa egy csúcsában a gyerekek számát a mágneslemez blokkmérete korlátozhatja.
- A futási időt két fő összetevőre bontjuk:
 - a lemezelérések számára, és
 - a központi egység (számolási) idejére.

B-fák

A T gyökeres fa **B-fa**, ha rendelkezik a következő tulajdonságokkal:

1. Minden x csúcsnak az alábbi mezői vannak:
 - $n[x]$ az x csúcsban tárolt kulcsok darabszáma,
 - az $n[x]$ darab kulcs, amelyeket nemcsökkenő sorrendben tárolunk:
 $kulcs_1[x] \leq kulcs_2[x] \leq \dots \leq kulcs_{n[x]}[x]$, és
 - $levél[x]$, amelynek értéke IGAZ, ha x levél és HAMIS, ha x egy belső csúcs.
2. Ha x egy belső csúcs, akkor tartalmazza a $c_1[x]$, $c_2[x]$, ..., $c_{n[x]+1}[x]$ mutatókat, amelyek x gyerekeire mutatnak. A levél csúcsoknak nincsenek gyerekeik, ezért a levelek c_i mezői definiálatlanok.
3. A $kulcs_i[x]$ értékek meghatározzák a kulcsértékeknek azokat a tartományait, amelyekbe a részfák kulcsai esnek. Ha k_i egy olyan kulcs, amelyik a $c_i[x]$ gyökerű részfában van, akkor:
 $k_1 \leq kulcs_1[x] \leq k_2 \leq kulcs_2[x] \leq \dots \leq kulcs_{n[x]}[x] \leq k_{n[x]+1}$.
4. Minden levélnek azonos a mélysége, ez az érték a fa h magassága.
5. A csúcsokban tárolható kulcsok darabszámára adott egy alsó és felső korlát. Ezeket a korlátokat egy rögzített t egész számmal ($t \geq 2$) lehet kifejezni, és ezt a számot a B-fa **minimális fokszámának** nevezzük:
 - Minden nem gyökér csúcsnak legalább $t-1$ kulcsa van. Minden nem gyökér belső csúcsnak legalább t gyereke van. Ha a fa nem üres, akkor a gyökércsúcsnak legalább egy kulcsa van.
 - Minden csúcsnak legfeljebb $2t-1$ kulcsa lehet, tehát egy belső csúcsnak legfeljebb $2t$ gyereke lehet. Azt mondjuk, hogy egy csúcs **telített**, ha pontosan $2t-1$ kulcsa van.

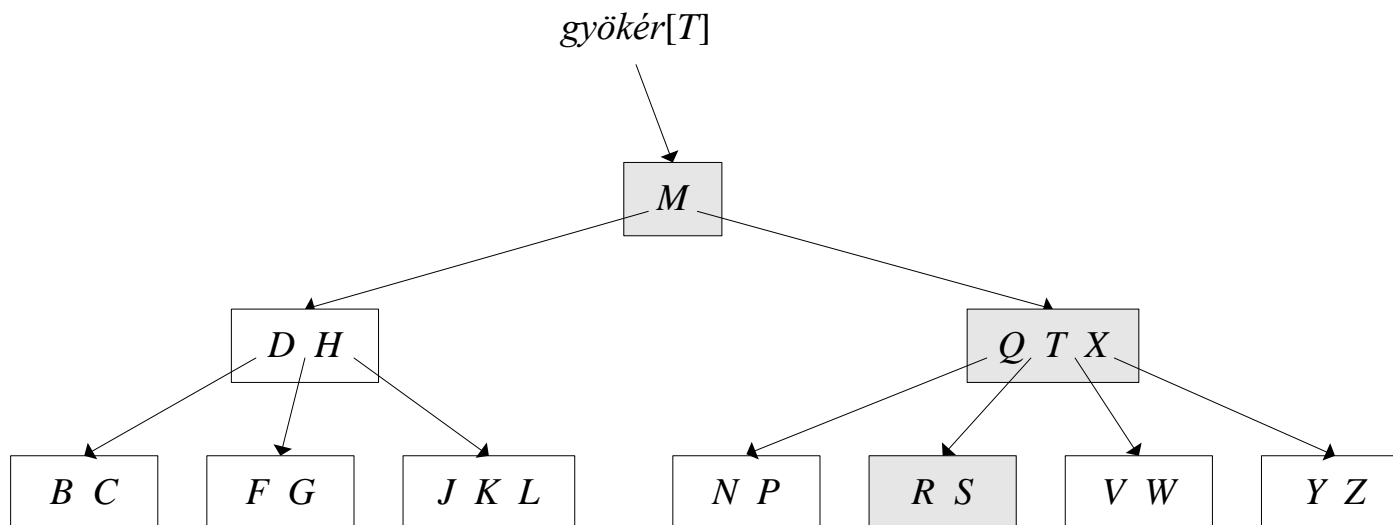
B-fák

Megjegyzés: A legegyszerűbb B-fára $t=2$. Ekkor minden belső csúcsnak 2, 3 vagy 4 gyereke van, ezért ezt a fát **2-3-4 fának** nevezzük. A gyakorlatban azonban ennél sokkal nagyobb t értékű fákat használnak.

Tétel: Ha $n \geq 1$ és T olyan n -kulcsos B-fa, amelynek magassága h és minimális fokszáma $t \geq 2$, akkor $h \leq \log_t ((n+1)/2)$.

Következmény: A legtöbb művelet a B-fák esetén legalább $(\lg t)$ -szer kevesebb csúcsot vizsgál meg, mint a piros-fekete fák esetén. Mivel egy csúcs vizsgálata rendszerint egy lemezhozzáférést igényel, ezért a lemezhozzáférések száma jelentősen csökken.

B-fák



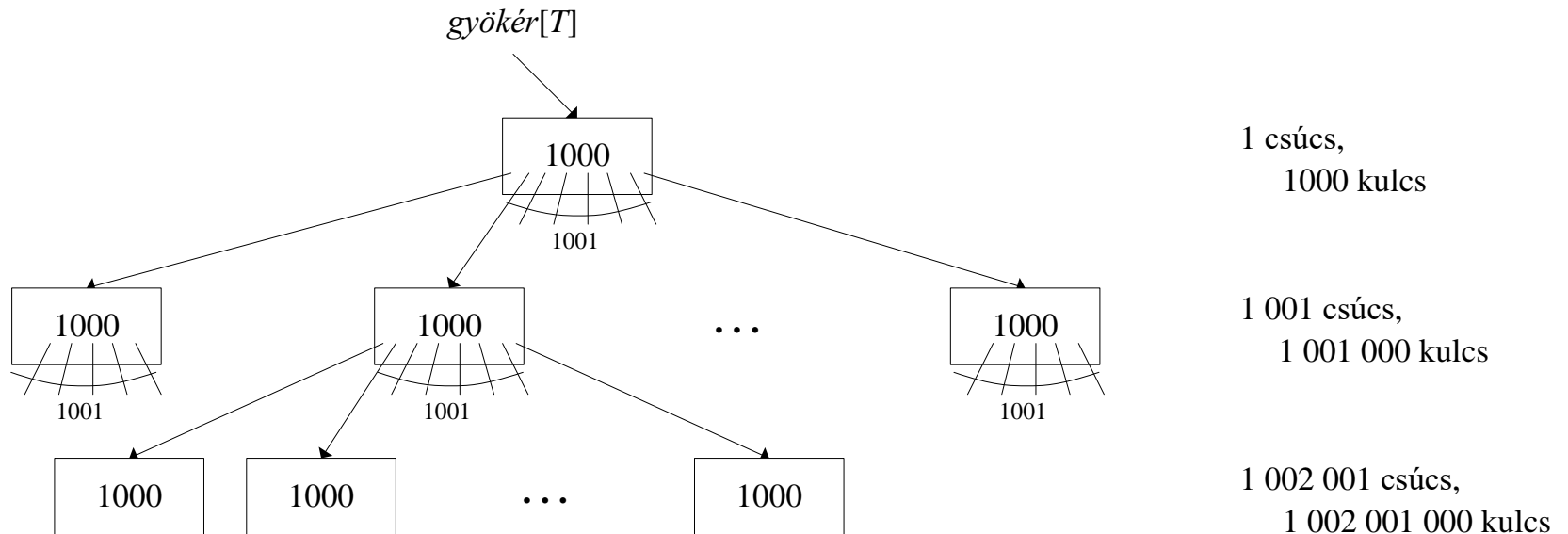
Egy B-fa

Feladatok

- Milyen t értékre lesz B-fa az előző dián szereplő fa?
- A minimális fokszám miért nem lehet 1?
- Adjuk meg az összes olyan B-fát, amelynek a minimális fokszáma 2, és az $\{1, 2, 3, 4, 5\}$ halmazt ábrázolják!
- Milyen az a 3 magasságú B-fa, amelyik a lehető legkevesebb kulcsot tartalmazza? A gyökércsúcs magassága 0.

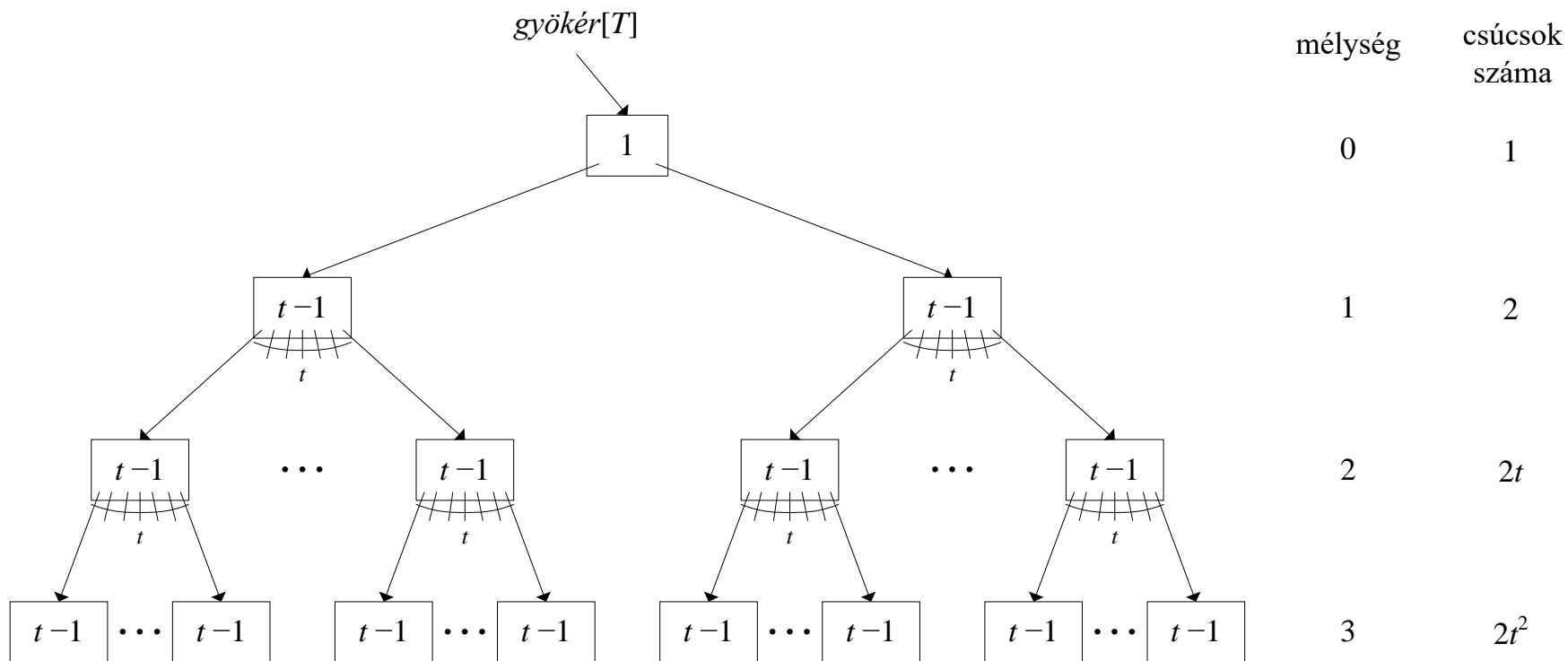


B-fák



Egy 2 magasságú B-fa több, mint egymilliárd kulccsal

B-fák



Egy 3 magasságú B-fa a lehető legkevesebb kulccsal

B-fák

A B-fák alapl műveleteinél feltesszük, hogy:

- A B-fa gyökere mindig a főmemóriában van, ezért a gyökércsúcsra a LEMEZRŐL-OLVAS műveletet sohasem kell végrehajtani, a LEMEZRE-ÍR művelet azonban kell, ha a gyökércsúcs megváltozott.
- Minden olyan csúcsra, amelyet paraméterként adunk át, már végrehajtottunk egy LEMEZRŐL-OLVAS műveletet.
- A CSÚCSOT-ELHELYEZ segéd eljárás $O(1)$ idő alatt lefoglalja az új csúcsnak a lemez egy blokkját. Nincs szükség a LEMEZRŐL-OLVAS meghívására, hiszen ehhez a csúcshoz semmilyen információt nem tároltunk még a lemezen.

B-fák

B-FÁBAN-KERES(x, k)

```

1   $i \leftarrow 1$ 
2  while  $i \leq n[x]$  és  $k > kulcs_i[x]$ 
3       $i \leftarrow i+1$ 
4  if  $i \leq n[x]$  és  $k = kulcs_i[x]$ 
5      return  $(x, i)$ 
6  if  $levél[x]$ 
7      return NIL
8  else
9      LEMEZRŐL-OLVAS( $c_i[x]$ )
10     return B-FÁBAN-KERES( $c_i[x], k$ )
    
```

Hatékonyság: $O(h)$ lemezművelet és $O(th)$ központi egység idő kell, ahol $h = \log_2 n$ a B-fa magassága, és n a kulcsok száma.

Feladatok

- Hogyan lehet egy B-fában a minimális kulcsot, és hogyan lehet egy adott kulcsot megelőző kulcsot megkeresni?



B-fák

B-FÁT-LÉTREHOZ(T)

- 1 $x \leftarrow \text{CSÚCSOT-ELHELYEZ}()$
- 2 $\text{levél}[x] \leftarrow \text{IGAZ}$
- 3 $n[x] \leftarrow 0$
- 4 $\text{LEMEZRE-ÍR}(x)$
- 5 $\text{gyökér}[T] \leftarrow x$

Hatékonyság: Az eljáráshoz $O(1)$ lemezművelet és $O(1)$ központi egység idő kell.

B-fák

```

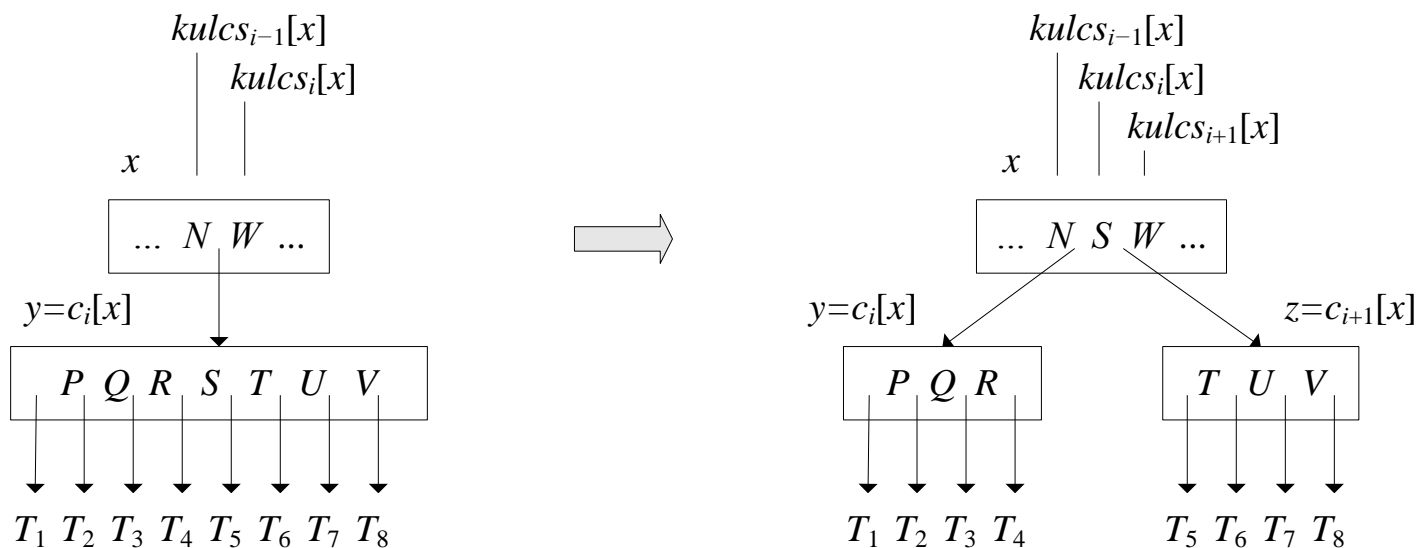
B-FA-VÁGÁS-GYEREK( $x, i, y$ )
1   $z \leftarrow \text{CSÚCSOT-ELHELVEZ}()$ 
2   $\text{levél}[z] \leftarrow \text{levél}[y]$ 
3   $n[z] \leftarrow t-1$ 
4  for  $j \leftarrow 1, t-1$ 
5       $\text{kulcs}_j[z] \leftarrow \text{kulcs}_{j+t}[y]$ 
6  if nem  $\text{levél}[y]$ 
7      for  $j \leftarrow 1, t$ 
8           $c_j[z] \leftarrow c_{j+t}[y]$ 
9   $n[y] \leftarrow t-1$ 
10 for  $j \leftarrow n[x]+1, i+1, -1$ 
11      $c_{j+1}[x] \leftarrow c_j[x]$ 
12  $c_{i+1}[x] \leftarrow z$ 
13 for  $j \leftarrow n[x], i, -1$ 
14      $\text{kulcs}_{j+1}[x] \leftarrow \text{kulcs}_j[x]$ 
15  $\text{kulcs}_i[x] \leftarrow \text{kulcs}_t[y]$ 
16  $n[x] \leftarrow n[x]+1$ 
17 LEMEZRE-ÍR( $y$ )
18 LEMEZRE-ÍR( $z$ )
19 LEMEZRE-ÍR( $x$ )
    
```

Hatékonyság: Az eljáráshoz $O(1)$ lemezművelet és $\Theta(t)$ központi egység idő kell.

Feltétel: Az x belső csúcs nem telített, az y az x telített gyereke, és $y = c_i[x]$.



B-fák



Egy csúcs szétvágása ($t = 4$)

B-fák

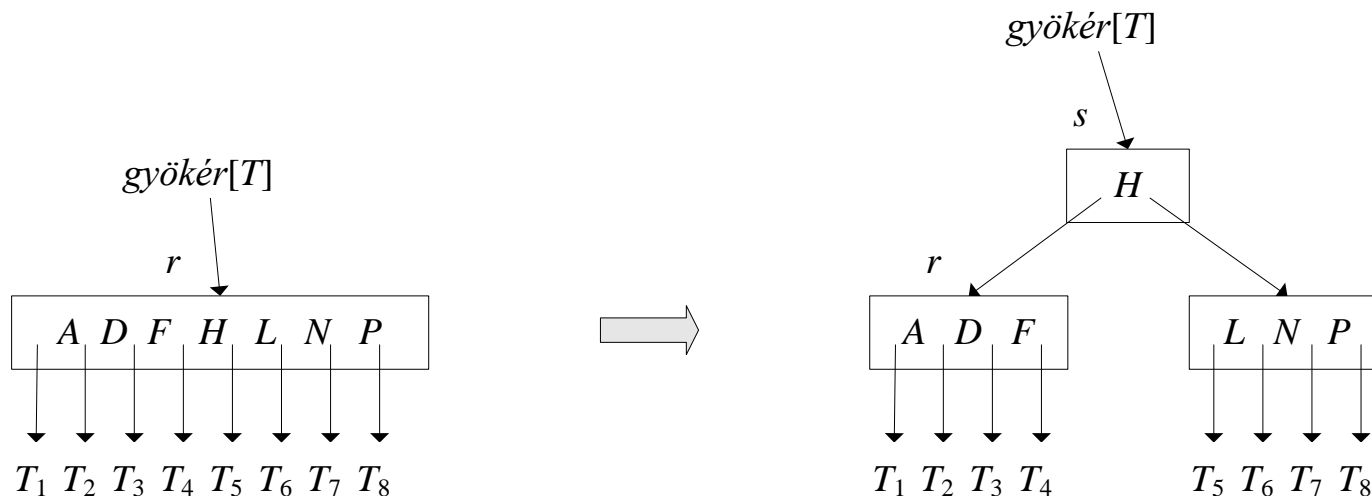
B-FÁBA-BESZÚR(T, k)

```

1   $r \leftarrow \text{gyökér}[T]$ 
2  if  $n[r] = 2t - 1$ 
3       $s \leftarrow \text{CSÚCSOT-ELHELYEZ}()$ 
4       $\text{gyökér}[T] \leftarrow s$ 
5       $\text{levél}[s] \leftarrow \text{HAMIS}$ 
6       $n[s] \leftarrow 0$ 
7       $c_1[s] \leftarrow r$ 
8      B-FA-VÁGÁS-GYEREK( $s, 1, r$ )
9      NEM-TELÍTETT-B-FÁBA-BESZÚR( $s, k$ )
10 else
11     NEM-TELÍTETT-B-FÁBA-BESZÚR( $r, k$ )
    
```

Hatékonyság: Az eljárás műveletigényét a meghívott eljárások műveletigénye adja, hiszen azok csak $O(1)$ lemezművelettel és $O(1)$ központi egység idővel bővülnek.

B-fák



A gyökércsúcs szétvágása ($t = 4$)

B-fák

NEM-TELÍTETT-B-FÁBA-BESZÚR(x, k)

```

1   $i \leftarrow n[x]$ 
2  if  $\text{levél}[x]$ 
3      while  $i \geq 1$  és  $k < \text{kulcs}_i[x]$ 
4           $\text{kulcs}_{i+1}[x] \leftarrow \text{kulcs}_i[x]$ 
5           $i \leftarrow i-1$ 
6       $\text{kulcs}_{i+1}[x] \leftarrow k$ 
7       $n[x] \leftarrow n[x]+1$ 
8      LEMEZRE-ÍR( $x$ )
9  else
10     while  $i \geq 1$  és  $k < \text{kulcs}_i[x]$ 
11          $i \leftarrow i-1$ 
12      $i \leftarrow i+1$ 
13     LEMEZRŐL-OLVAS( $c_i[x]$ )
14     if  $n[c_i[x]] = 2t-1$ 
15         B-FA-VÁGÁS-GYEREK( $x, i, c_i[x]$ )
16         if  $k > \text{kulcs}_i[x]$ 
17              $i \leftarrow i+1$ 
18     NEM-TELÍTETT-B-FÁBA-BESZÚR( $c_i[x], k$ )
    
```

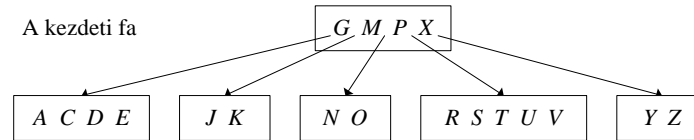
Hatékonyság: $O(h)$ lemezművelet és $O(th)$ központi egység idő kell, ahol $h = \log_2 n$ a B-fa magassága.

Feltétel: Az x csúcs nem telített.

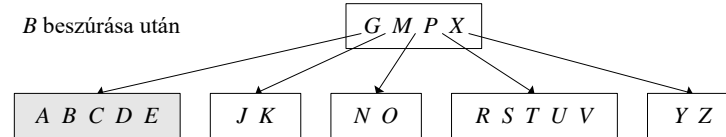


B-fák

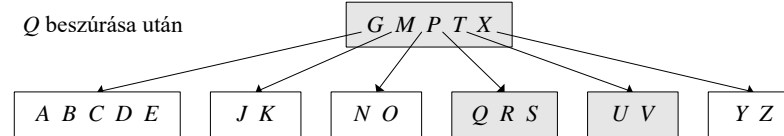
A kezdeti fa



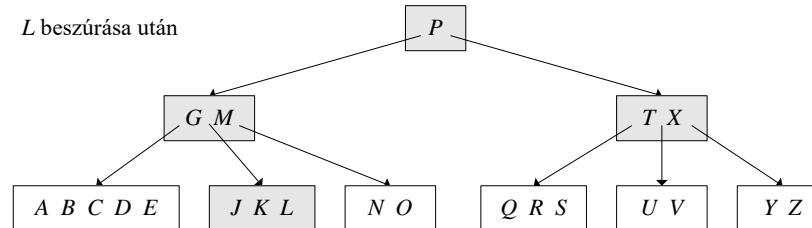
B beszúrása után



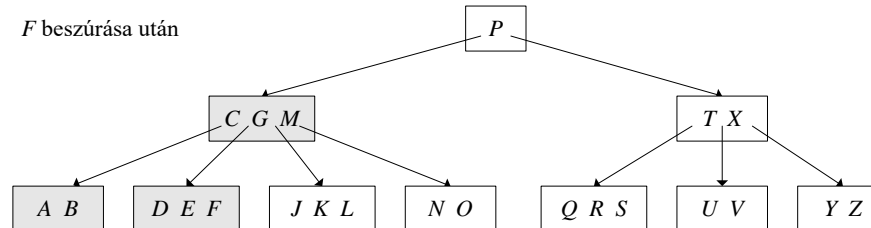
Q beszúrása után



L beszúrása után



F beszúrása után



Kulcsok beszúrása egy B-fába ($t = 3$)

Feladatok

- Milyen B-fát kapunk eredményül, ha egy kezdetben üres B-fába az alábbi kulcsokat szűrjük? A B-fa minimális fokszáma legyen 3.
 - *F, S, Q, K, C, L, H, T, V, W, M, R, N, P, A, B, X, Y, D, Z, E*
- Milyen B-fát kapunk eredményül akkor, ha a B-fa minimális fokszáma 2?

