Bináris fák
2. előadás

(Horváth Gyula anyagai felhasználásával)

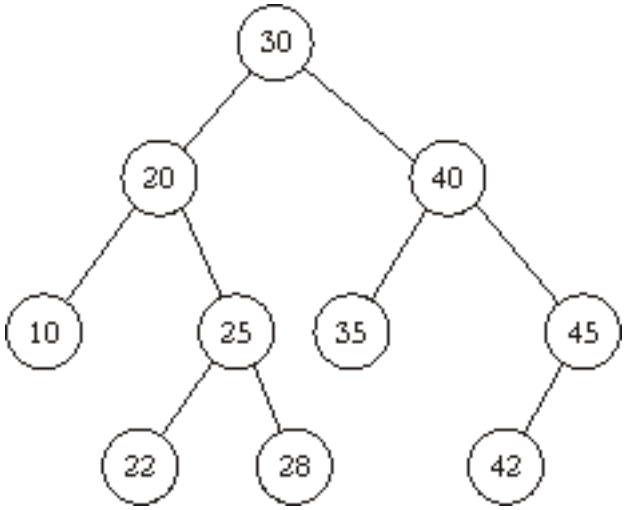
Kereső- és rendezőfák

Közös tulajdonságok:

- A gyökérelem (vagy kulcsértéke) nagyobb vagy egyenlő minden tőle balra levő elemnél.
- A gyökérelem (vagy kulcsértéke) kisebb vagy egyenlő minden tőle jobbra levő elemnél.

Keresőfa specialitása:

- Nincs két azonos (kulcsú) elem.



Kereső- és rendezőfák

A keresőfa új műveletei:

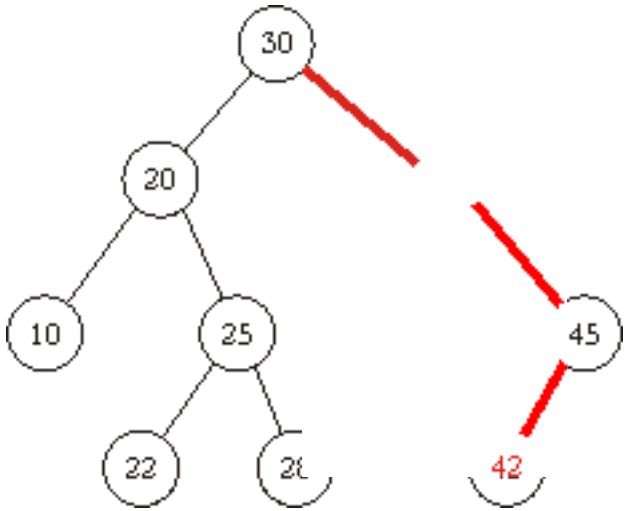
- keresés
- beillesztés
- törlés
- kiegyensúlyozás

Megjegyzés: Ha egy halmazra az *eleme?*, *elem hozzávétele*, illetve *elem törlése* művelet szükséges, akkor a halmazt ábrázolhatjuk keresőfával – azaz megkaptuk a halmaz típus egy újabb ábrázolását!

Ezzel szemben az unió és a metszet művelet nehéz!

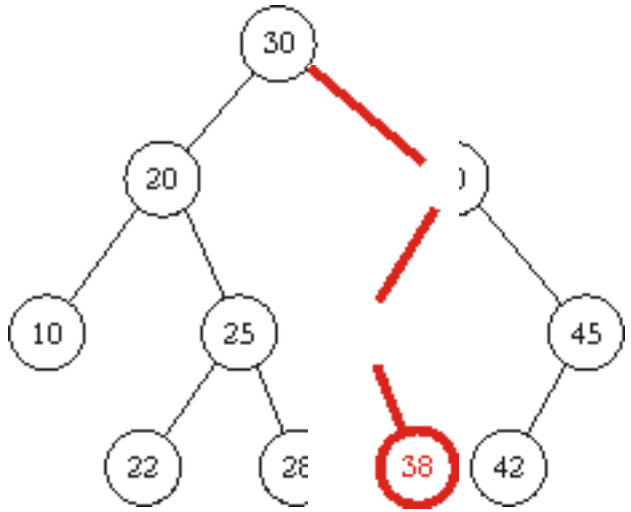
Kereső- és rendezőfák

```
Keresés(k,kf):
  Elágazás
    Üres?(kf)esetén Keresés:=kf
    k<gyökérelem(kf).kulcs
       esetén Keresés:=Keresés(k,balgyerek(kf))
    k>gyökérelem(kf).kulcs
       esetén Keresés:=Keresés(k,jobbgyerek(kf))
    k=gyökérelem(kf).kulcs
       esetén Keresés:=kf
  Elágazás vége
Függvény vége.
```



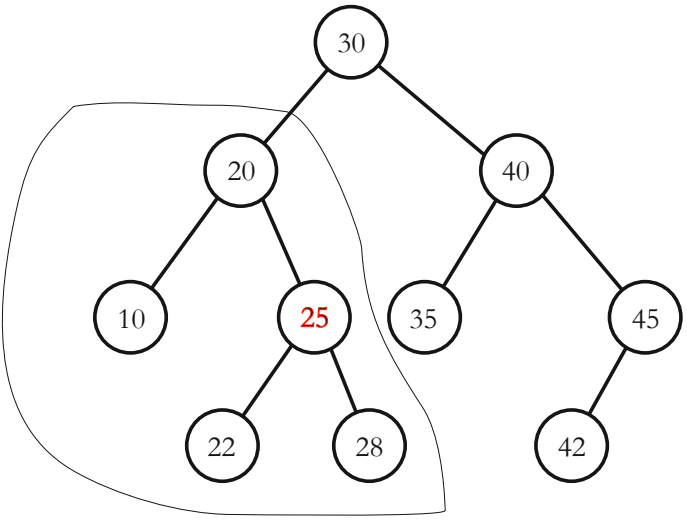
Kereső- és rendezőfák

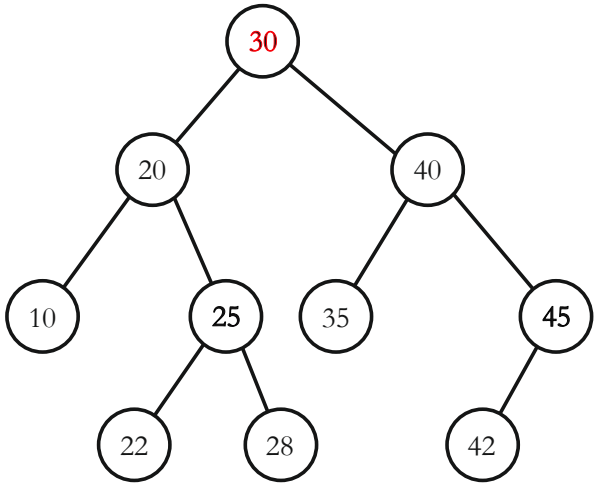
```
Beillesztés(e,kf):
  Elágazás
    Üres?(kf)esetén kf:=egyeleműfa(e)
    e.kulcs<gyökérelem(kf).kulcs
       esetén Beillesztés(e,balgyerek(kf))
    e.kulcs>gyökérelem(kf).kulcs
       esetén Beillesztés(e,jobbgyerek(kf))
  Elágazás vége
Függvény vége.
```

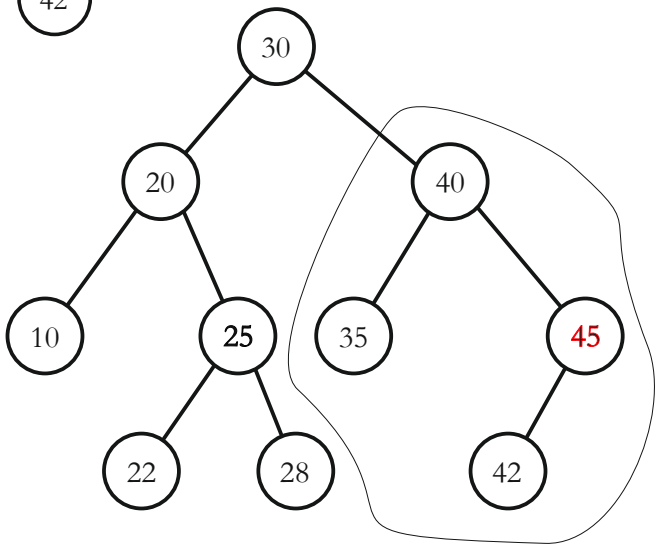


Kereső- és rendezőfák

Törlés







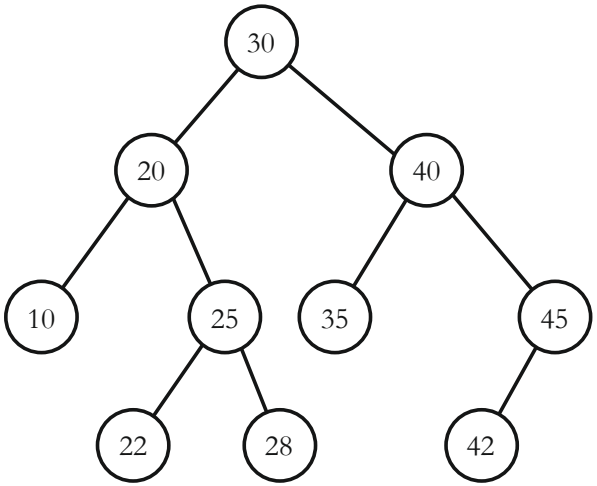
Kereső- és rendezőfák

```
Törlés(k,kf):
  Elágazás
    k<gyökérelem(kf).kulcs
       esetén Törlés(k,balgyerek(kf))
    k>gyökérelem(kf).kulcs
       esetén Törlés(k,jobbgyerek(kf))
    k=gyökérelem(kf).kulcs
       esetén Gyökértörlés(k,kf)
  Elágazás vége
Függvény vége.
```

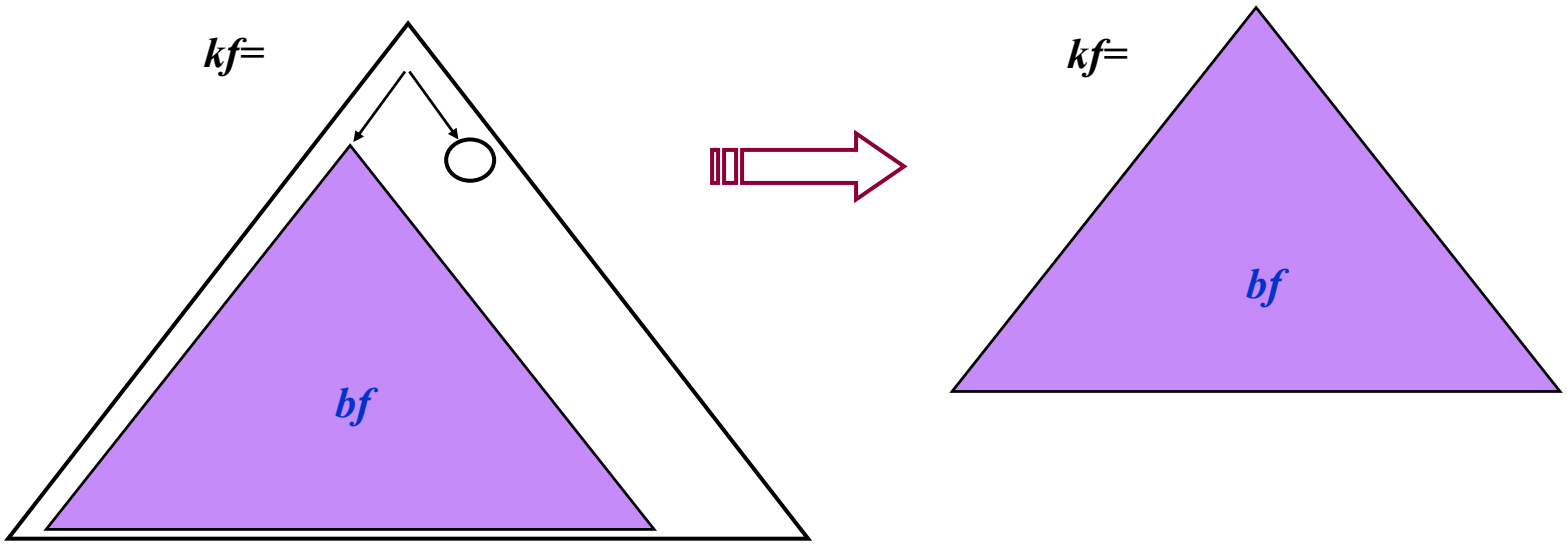
Kereső- és rendezőfák

A megoldandó estek: a törlendő

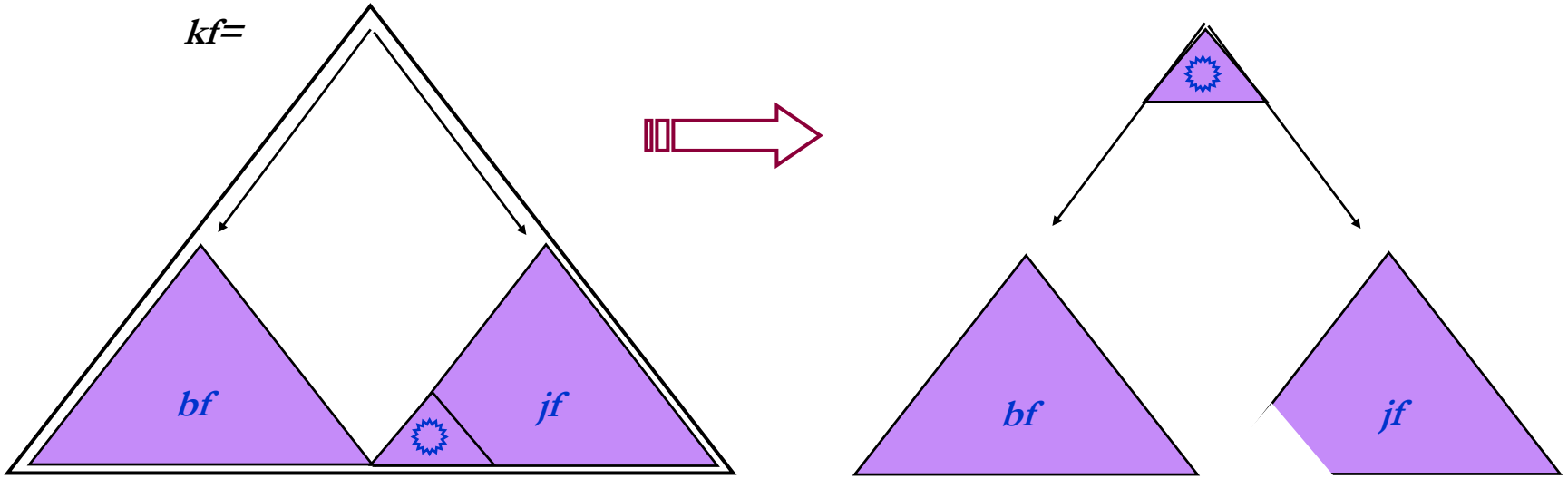
- levélelem
- nincs jobboldali részfája
- nincs baloldali részfája
- egyik részfája sem üres



Kereső- és rendezőfák



Kereső- és rendezőfák



Kereső- és rendezőfák

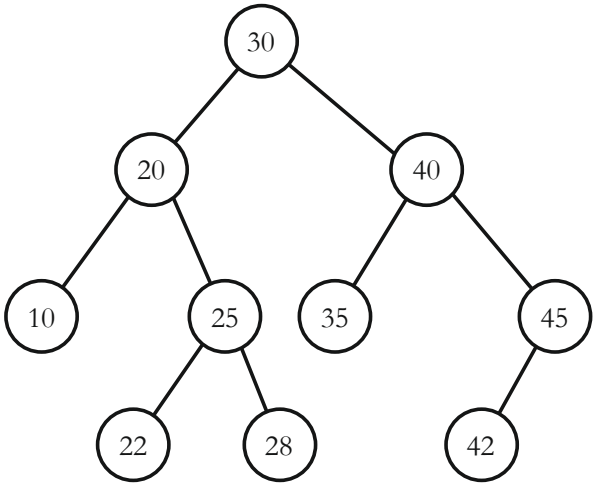
```
Gyökértörlés(k,kf):
  c:=kf
  Ha üres?(balgyerek(kf))
     akkor kf:=jobbgyerek(kf)
  különben ha üres?(jobbgyerek(kf))
     akkor kf:=balgyerek(kf)
  különben Legbal(kf,jobbgyerek(kf),lb)
           Gyökérmódosít(kf,lb)
  Elágazás vége
  Felszabadít(c)
Függvény vége.
```

Lehetne a másik oldalról is:

```
Legjobb(balgyerek(kf),lj)
```

Kereső- és rendezőfák

```
Legbal(szülő,kf,lb):
  Ha nem üres?(balgyerek(kf))
     akkor Legbal(kf,balgyerek(kf),lb)
  különben lb:=gyökérelem(kf); t:=kf
            Balrailleszt(szülő,jobbgyerek(kf))
            Felszabadít(t)
Függvény vége.
```

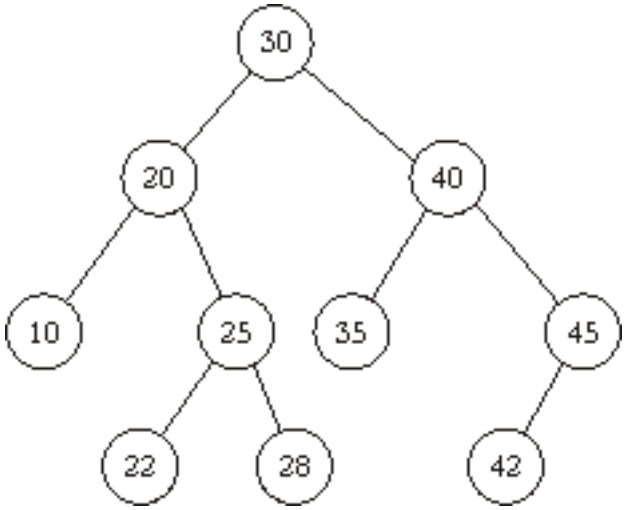


Kereső- és rendezőfák keresés

A keresőfa műveletei módosítása:

- legkisebb elem – a fa legbaloldalibb eleme

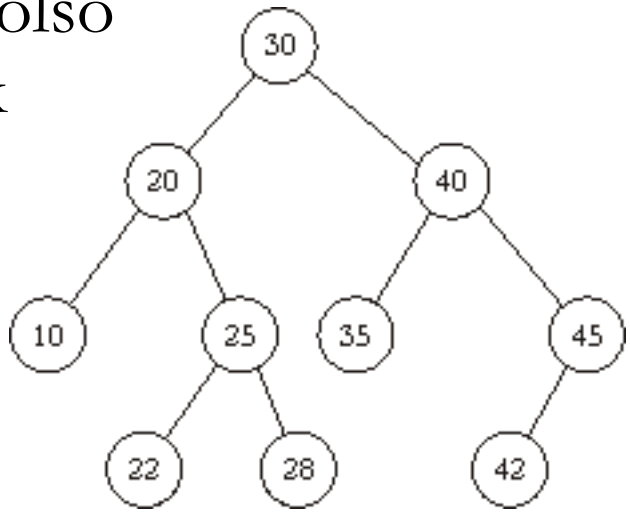
```
minimum(kf):
   p:=kf
   Ciklus amíg nem üres?(balgyerek(p))
     p:=balgyerek(p)
   Ciklus vége
   minimum:=p
Függvény vége.
```



Kereső- és rendezőfák keresés

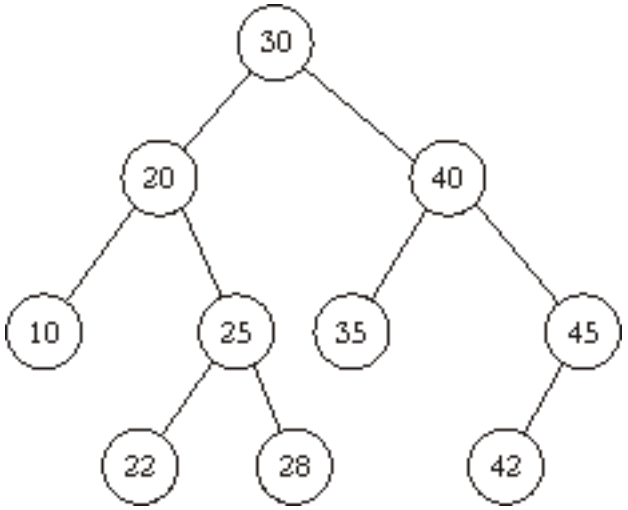
A keresőfa műveletei módosítása:

- Adott elemet követő elem (ismerni kell minden elemre a szülőt is):
  - Ha van jobb-gyereke, akkor a jobboldali ág legbaloldalibb eleme (pl. 30 → 35).
  - Ha nincs jobb-gyereke, akkor az utolsó szülő, ahol keresésnél balra léptünk (pl. 28 → 30).



Kereső- és rendezőfák keresés

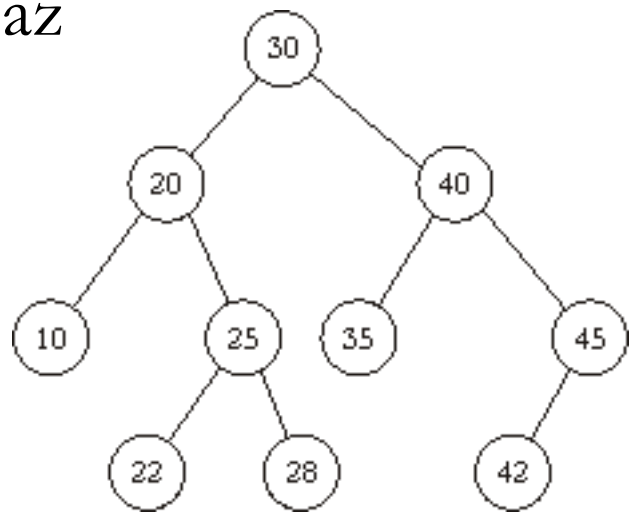
```
következő(kf):
 Ha nem üres?(jobbgyerek(kf))
    akkor következő:=minimum(jobbgyerek(kf))
 különben s:=szülő(kf)
          Ciklus amíg s≠sehova és kf=jobbgyerek(s)
           kf:=s; s:=szülő(s)
          Ciklus vége
          következő:=s
Függvény vége.
```



Kereső- és rendezőfák keresés

A keresőfa műveletei módosítása:

- Adott értéket követő elem (ismerni kell minden elemre a szülőt is):
  - Ha balra lépünk ki a fából, akkor amelyik elemből kiléptünk (pl. 21 → 22).
  - Ha jobbra lépünk ki a fából, akkor az utolsó szülő, ahol keresésnél balra léptünk (pl. 29 → 30).

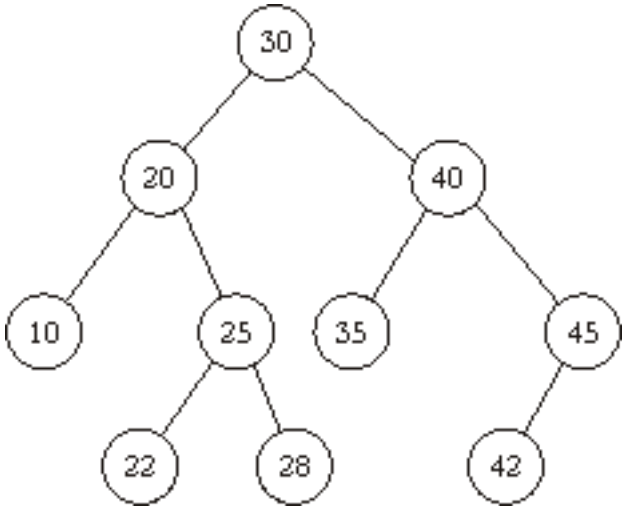


Kereső- és rendezőfák kiválogatás

A és B közötti összes elem kiválogatása egy sorba, növekvően.

Esetek (a gyökérelemhez képest):

- `10..25`
- `15..30`
- `40..50`
- `30..45`
- `20..40`

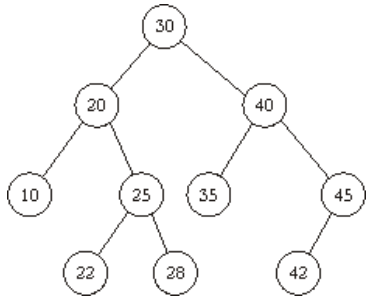


Kereső- és rendezőfák kiválogatás

A keresőfa műveletei módosítása:

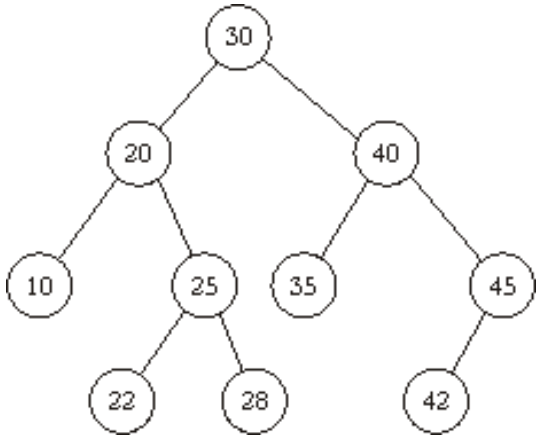
- A és B közötti összes elem kiválogatása egy sorba, növekvően.

```
Kiválogatás(a,b,kf):
 Ha nem üres?(kf) akkor
  Ha b<gyökérelem(kf).kulcs akkor
     Kiválogatás(a,b,balgyerek(kf))
  különben ha a>gyökérelem(kf).kulcs akkor
     Kiválogatás(a,b,jobbgyerek(kf))
  különben ha b=gyökérelem(kf).kulcs akkor
     Kiválogatás(a,b,balgyerek(kf))
     Sorba(gyökérelem(kf))
                  ...
```



Kereső- és rendezőfák kiválogatás

```
  különben ha a=gyökérelem(kf).kulcs akkor
    Sorba(gyökérelem(kf))
    Kiválogatás(a,b,jobbgyerek(kf))
  különben
    Kiválogatás(a,gyökérelem(kf).kulcs,
                balgyerek(kf))
    Sorba(gyökérelem(kf))
    Kiválogatás(gyökérelem(kf).kulcs,b,
                jobbgyerek(kf))
 Elágazások vége
Eljárás vége.
```



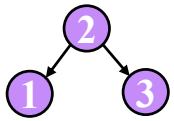
Kereső- és rendezőfák

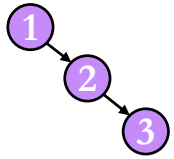
Kiegyensúlyozás – definíciók:

1. **Kiegyensúlyozott** az a bináris fa, amelynek tetszőleges pontjában „gyökerező” részfáinak águhosszai legfeljebb eggyel térnek el egymástól (AVL-fa).

2. **Teljesen kiegyensúlyozott** az a bináris fa, amelynek tetszőleges pontjában „gyökerező” részfáinak pontszámai legfeljebb eggyel térnek el egymástól.

3. **Teljes** az a bináris fa, amelynek tetszőleges pontjában „gyökerező” részfáinak pontszámai pontosan megegyeznek.

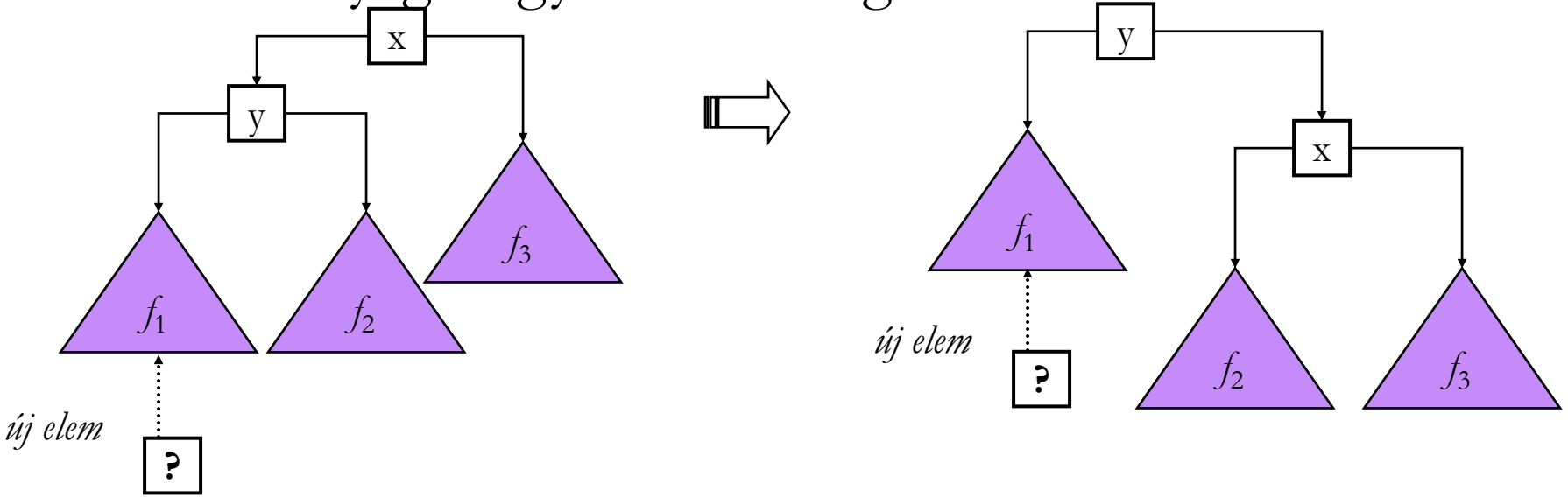




Kereső- és rendezőfák

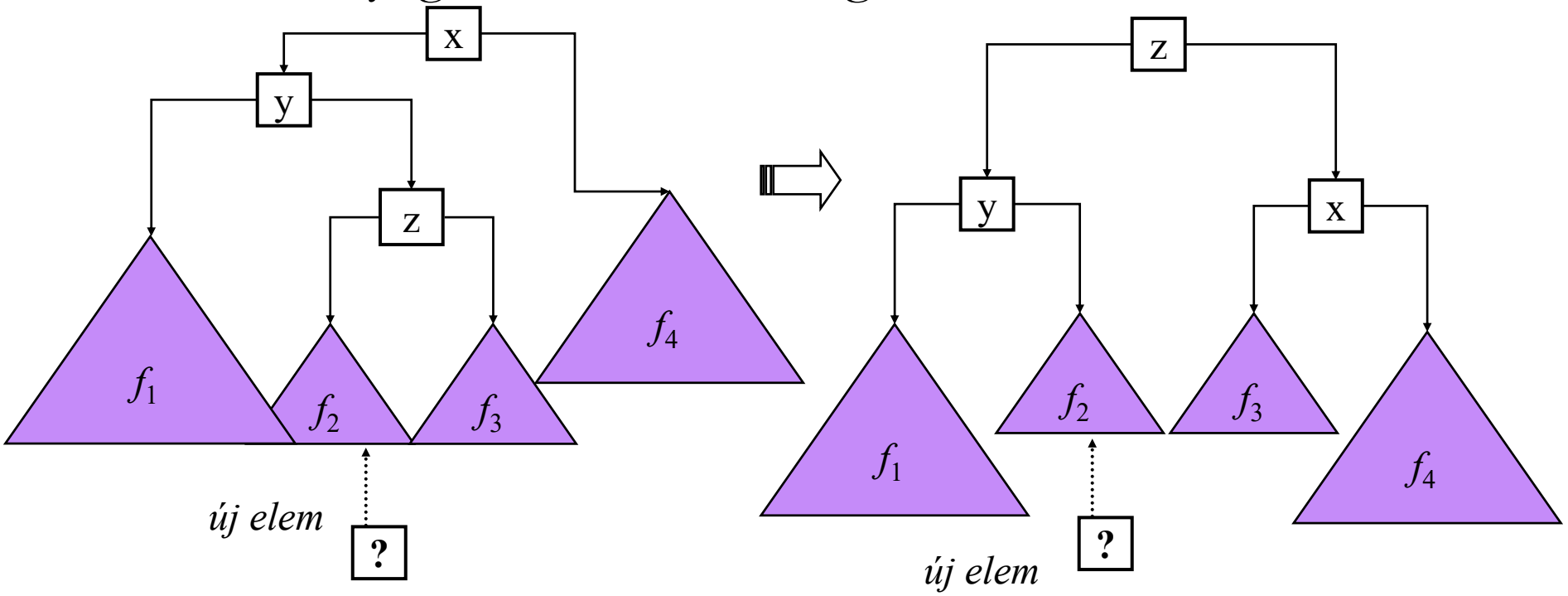
Cél: a beszúrás és a törlés műveletek módosítása úgy, hogy ha előtte az AVL-tulajdonság teljesült, akkor utána is teljesüljön.

A beszúrás lényege: egyszeres elforgatás



Kereső- és rendezőfák

A beszúrás lényege: kétszeres elforgatás



Kereső- és rendezőfák

A **piros-fekete fa** olyan bináris keresőfa, melyre :

- minden nem levél csúcsnak 2 gyereke van;
- elemeket belső csúcsokban tárolunk;
- a fa minden csúcsa piros vagy fekete;
- a gyökér és a levelek feketék;
- minden piros csúcs mindkét gyereke fekete
- minden csúcsra levélbe vezető utakon ugyanannyi fekete csúcs van.

Ha egy piros-fekete fában n elemet tárolunk, akkor a magassága $\leq 2 \log(n + 1)$.

Kereső- és rendezőfák

Kérdés: Lehet-e olyan keresőfa, amelyben több azonos értékű (kulcsú) elem is van?

A megoldáshoz egy szabályt módosítani kell: az ilyen keresőfa minden elemétől balra csak nála kisebbek, jobbra nála nagyobb vagy egyenlők lehetnek.

A keresés ekkor az azonos értékűekből az elsőt találja meg.

Szükség van egy új műveletre: az aktuálisat (utoljára megtaláltat) követő elem megadására.

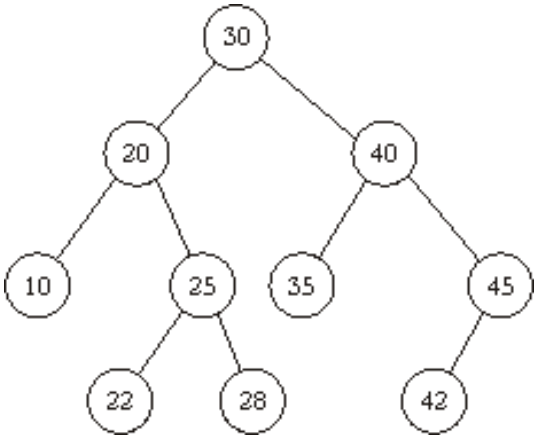
Ha a tárolási szabályt nem módosítjuk, akkor még az aktuálisat megelőző elem megadására is szükségünk lehet.

Kereső- és rendezőfák

A **rendezőfa** új műveletei:

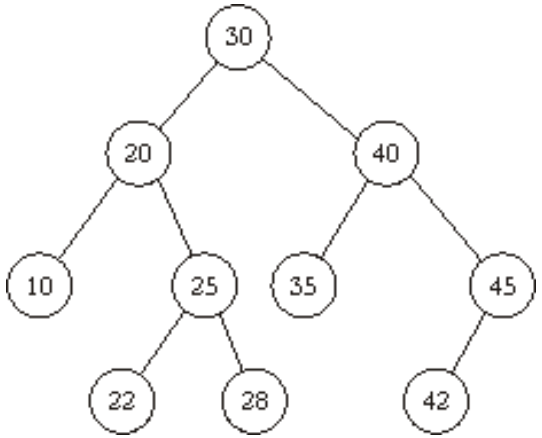
- beillesztés (majdnem azonos a keresőfával – de itt lehetnek egyforma értékű elemek)
- BKJ-bejárás

Megjegyzés: Itt törlés művelet nincs, a fát egyszer építjük fel, bejárjuk, majd a teljes fát törölhetjük.



Kereső- és rendezőfák

```
Beillesztés(e,kf):
  Elágazás
    Üres?(kf) esetén kf:=egyeleműfa(e)
    e.kulcs≤gyökérelem(kf).kulcs
       esetén Beillesztés(e,balgyerek(kf))
    e.kulcs>gyökérelem(kf).kulcs
       esetén Beillesztés(e,jobbgyerek(kf))
  Elágazás vége
Függvény vége.
```



Bináris fa – halmazból a K. legnagyobb eltávolítása

Halmazból a K. legnagyobb eltávolítása:

Néhány feladatnál (pl. permutációk előállítása inverziós táblával) felmerül, hogy egy halmazból ki kell venni a K. legnagyobb elemet.

Ha a halmazt rendezett sorozatként (tömb) képzeljük el, akkor a K. legnagyobb megtalálása egyetlen képlet kiszámítása, a kihagyása azonban a halmaz elemszámával arányos időben történik – el kell tolni előre a mögötte levő elemeket.

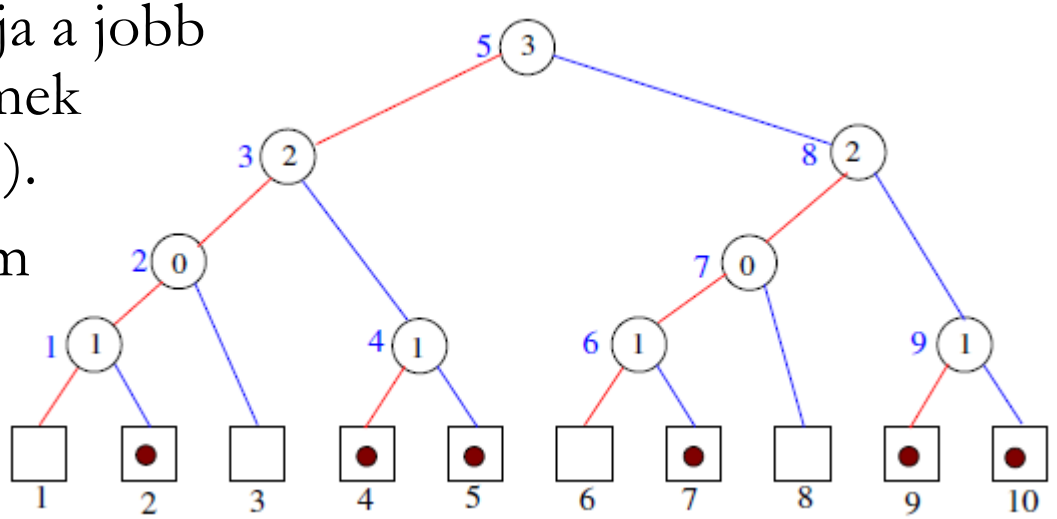
Ha a halmazt listával ábrázolnánk, akkor pedig a K. legnagyobb megtalálási ideje arányos a halmaz elemszámával.

Az ötlet: ábrázoljuk a halmazt bináris fával!

Bináris fa – halmazból a K. legnagyobb eltávolítása

1. A fának n levélpontja van.
2. A fának n-1 belső (nem levél) pontja van az 1..n-1 számokkal jelölve.
3. Minden belső pontnak két fia van.
4. Az a..b elemeket (leveleket) tartalmazó részfa gyökere $k = \lfloor (a+b)/2 \rfloor$, bal részfája az a..k, jobb részfája pedig a k+1..b elemeket tartalmazza.
5. A fa minden p belső pontja a jobb részfájában lévő halmaz-elemek számát tartalmazza (Hány(p)).

Legyen H(i) igaz, ha az i elem még a halmazban van!

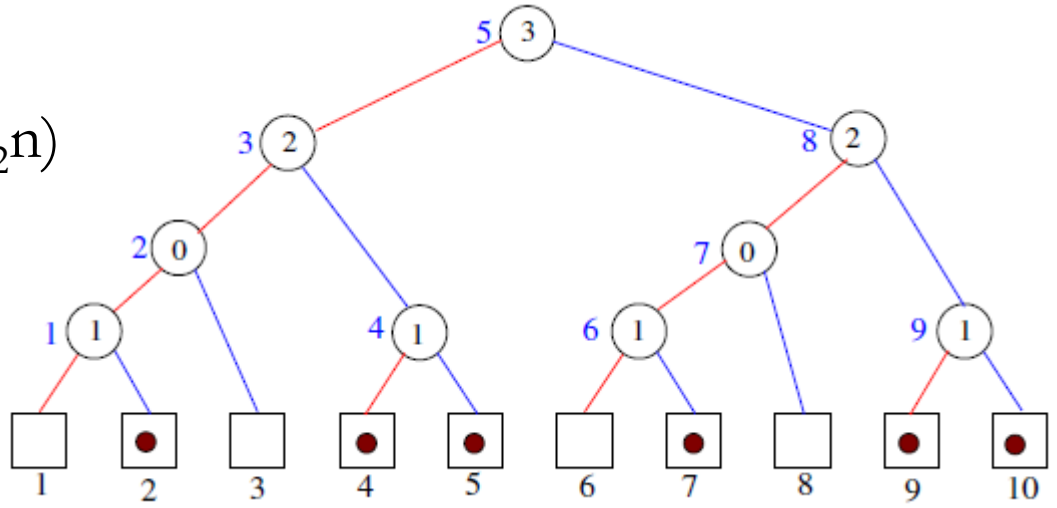


A {2;4;5;7;9;10} halmaz ábrázolása, ha n=10.

Bináris fa – halmazból a K. legnagyobb eltávolítása

```
Keres(k,bal,jobb):
 Ha bal=jobb akkor H(bal):=hamis; Keres:=bal
 különben
   s=(bal+jobb)/2
   Ha Hány(s)>=k akkor Hány(s):=Hány(s)-1
                       Keres:=Keres(k,s+1,jobb)
   különben Keres:=Keres(k-Hány(s),bal,s)
 Elágazások vége
Függvény vége.
```

Ez a `Keres` művelet $O(\log_2 n)$ futási idejű.

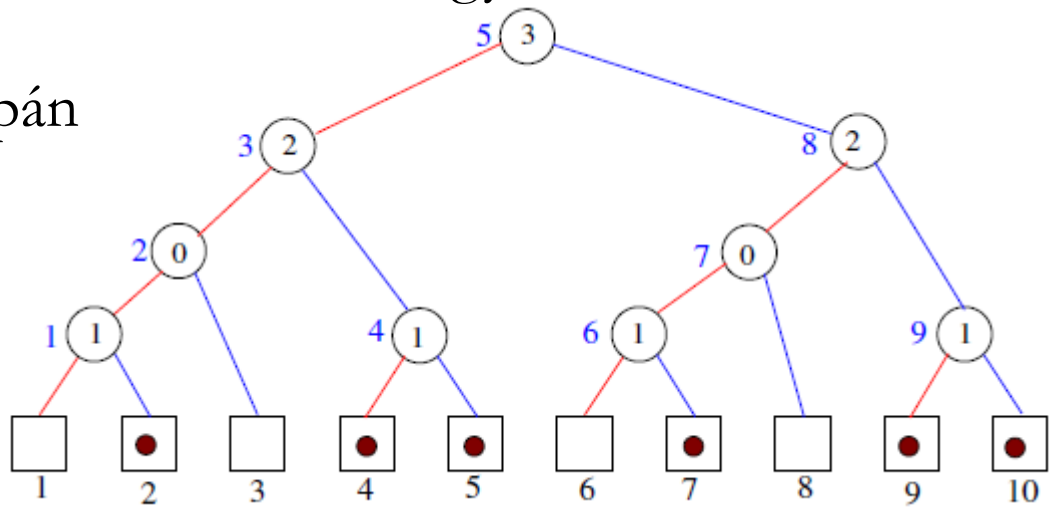


Bináris fa – halmazból a K. legnagyobb eltávolítása

A bináris fa „létrehozása” n elemből: (`Épít(1,n)`)

```
Épít(bal,jobb):
  Ha bal<jobb akkor s=(bal+jobb)/2
                    Hány(s):=jobb-s
                    Épít(bal,s); Épít(s+1,jobb)
Eljárás vége.
```

A keresés és az építés műveletekből is látszik, hogy itt a bináris fát ténylegesen nem hozzuk létre, (ahogyan a kupacnál sem) csupán modellként használjuk a feladat megoldásában.

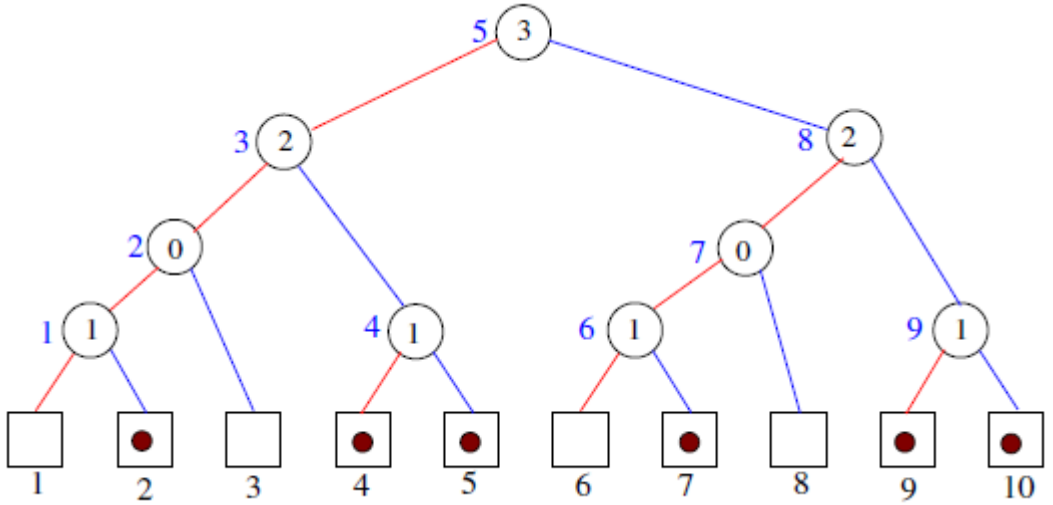


Bináris fa – halmazból a K. legnagyobb eltávolítása

A bináris fa „létrehozása” adott H-beli elemekből:

```
Épít(bal,jobb,db):
  Ha bal<jobb
     akkor s=(bal+jobb)/2
           Épít(bal,s,b); Épít(s+1,jobb,j)
           Hány(s):=j; db:=b+j
  különben Ha H(bal) akkor db:=1 különben db:=0
Eljárás vége.
```

```
Épít(1,n,db)
```



Szegmens fa

Feladat: Határozzuk meg egy `X` vektor `i`. és `j`. eleme között levő elemei minimumát!

A feladat felfogható egy egyszerű minimum-kiválasztás tétel alkalmazásként, ahol az eredmény meghatározásához `j-i` összehasonlítást kell végezni, azaz a futási idő a vektor elemszámával arányos.

A feladat azonban megoldható másként is, egy speciális bináris fa, a szegmens fa alkalmazásával. A bináris fa elkészítése is munkaigényes, de gazdaságos, ha sok intervallumon kell minimumot keresnünk.

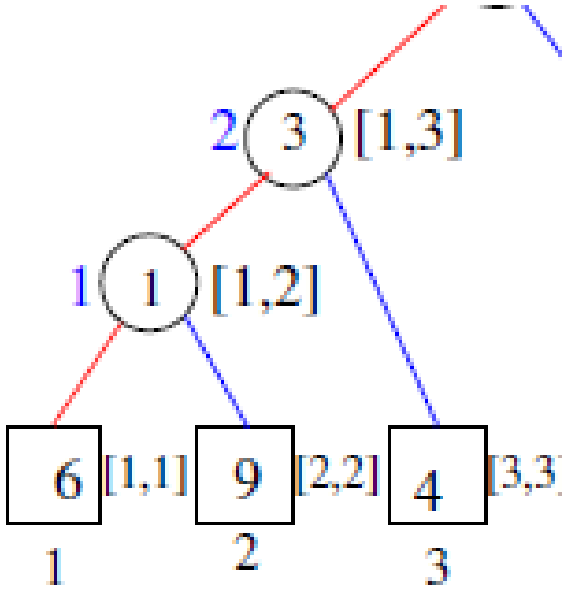
Szegmens fa

A szegmens fa levelei a tömb elemeit tartalmazzák.

A fa többi elemének indexét a hozzá tartozó két szélső levélelem indexének átlagaként számoljuk.

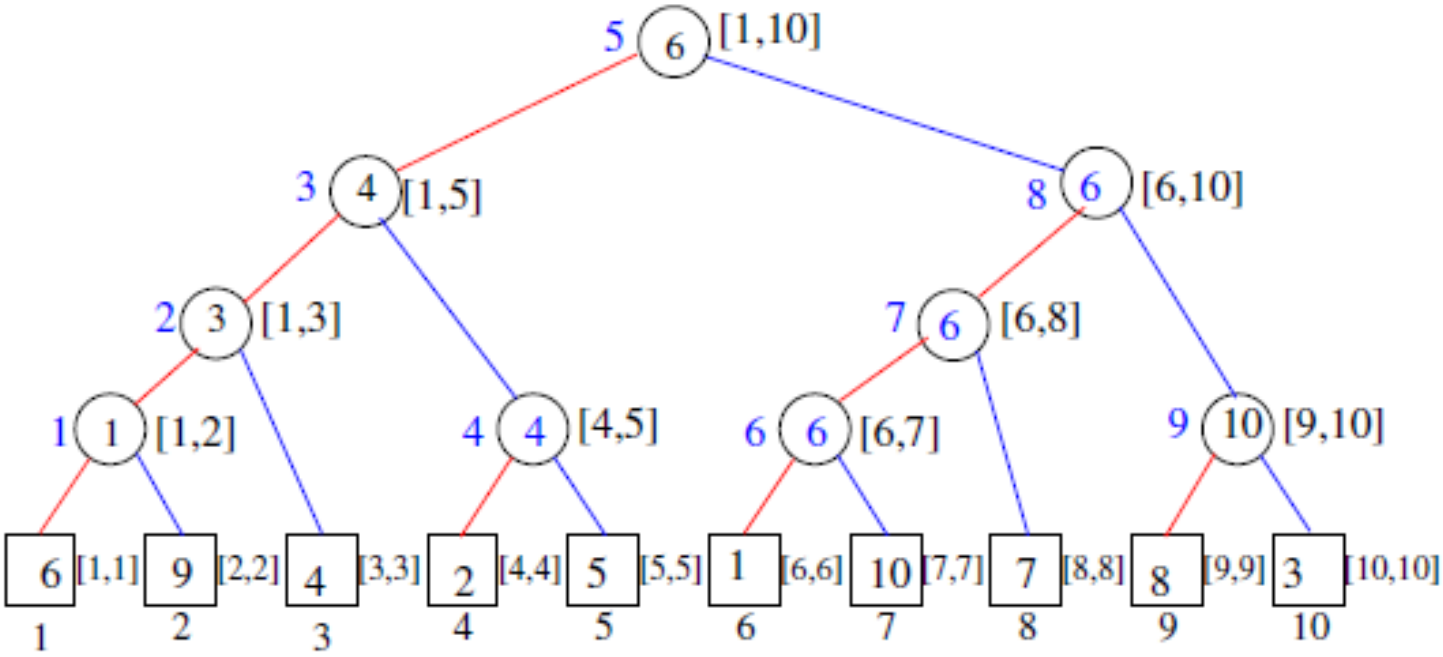
Minden belső elemhez rendeljük hozzá azt az eredeti tömbindexet, ahol a levelei minimuma található!

Mint látható, ez a fa is egy modell, állandó szerkezetű, a megfelelő indexek egy tömbben tárolhatók.



Szegmens fa

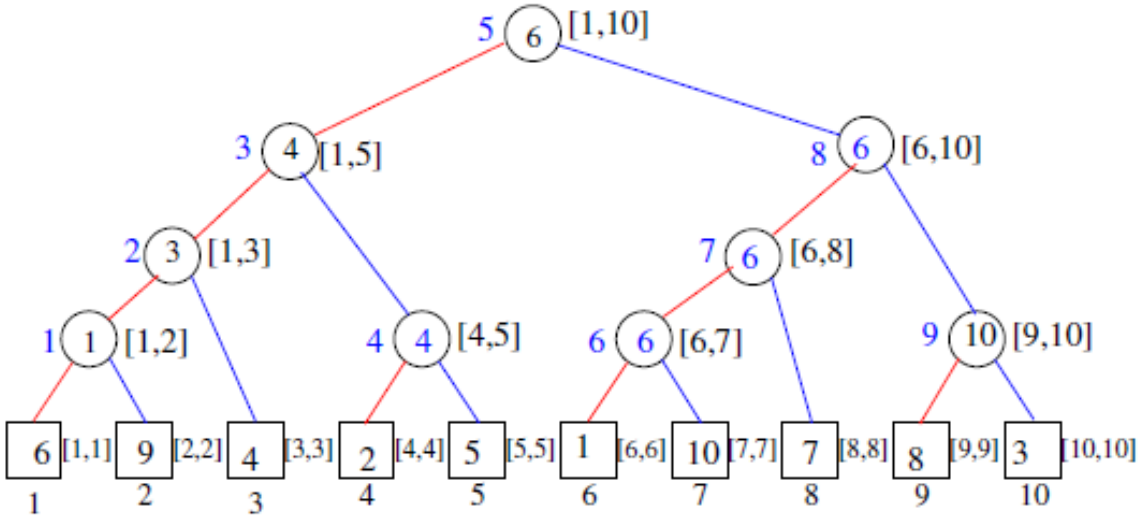
Minden belső elemhez rendeljük hozzá azt az eredeti tömbindexet, ahol a levelei minimuma található!



Szegmens fa a (6;9;4;2;5;1;10;7;8;3) sorozatra.

Szegmens fa

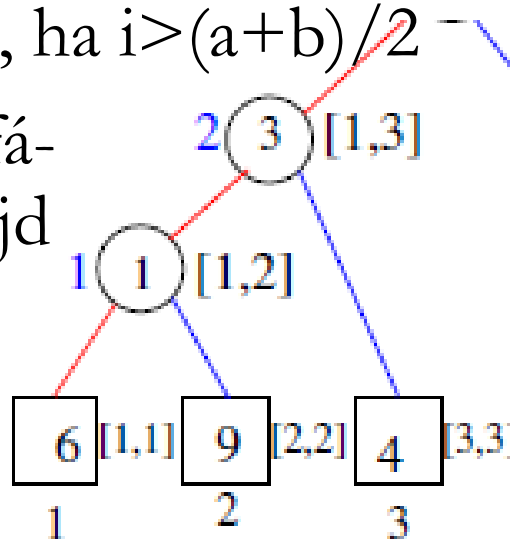
```
Felépít(a,b):
   Ha a=b akkor Felépít:=a
   különben k:=(a+b)/2
            x:=Felépít(a,k); y:=Felépít(k+1,b)
            Ha A(x)<A(y) akkor Fa(k):=x
                         különben Fa(k):=y
            Felépít:=Fa(k)
Függvény vége.
```



Szegmens fa

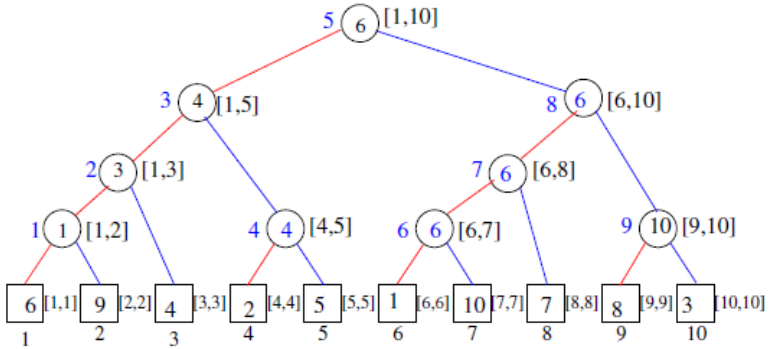
Ezután az [a,b] szegmensfában az [i,j] intervallum minimumának meghatározása rekurzívan megadható:

- i, ha i=j vagy A(i)
- FA((a+b)/2), ha a=i és j=b vagy A(FA((a+b)/2))
- [a,(a+b)/2] szegmensfában [i,j] minimuma, ha j≤(a+b)/2
- [(a+b)/2+1,b] szegmensfában [i,j] minimuma, ha i>(a+b)/2
- [i,j]-t felosztjuk a bal- és jobboldali szegmensfára, mindkettőben minimumot keresünk, majd vesszük a két elem minimumát.



Szegmens fa

```
Keres(a,b,i,j):
   k:=(a+b)/2
   Ha i=j akkor Keres:=i
   különben ha a=i és b=j akkor Keres:=FA(k)
   különben ha j≤k akkor Keres:=Keres(a,k,i,j)
   különben ha i>k akkor Keres:=Keres(k+1,b,i,j)
   különben x:=Keres(a,k,i,k)
            y:=Keres(k+1,b,k+1,j)
            Ha A(x)<A(y) akkor Keres:=x
                         különben Keres:=y
Függvény vége.
```



Bináris fák
2. előadás vége