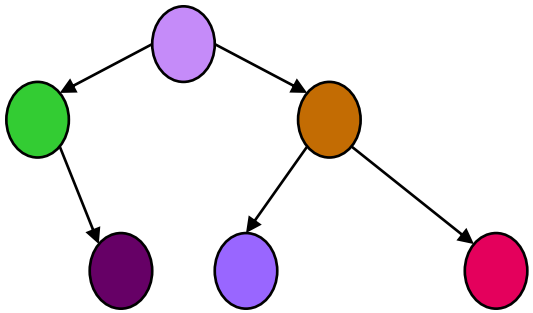
Fák, bináris fák

Bináris fa

A fa (bináris fa) rekurzív adatszerkezet:

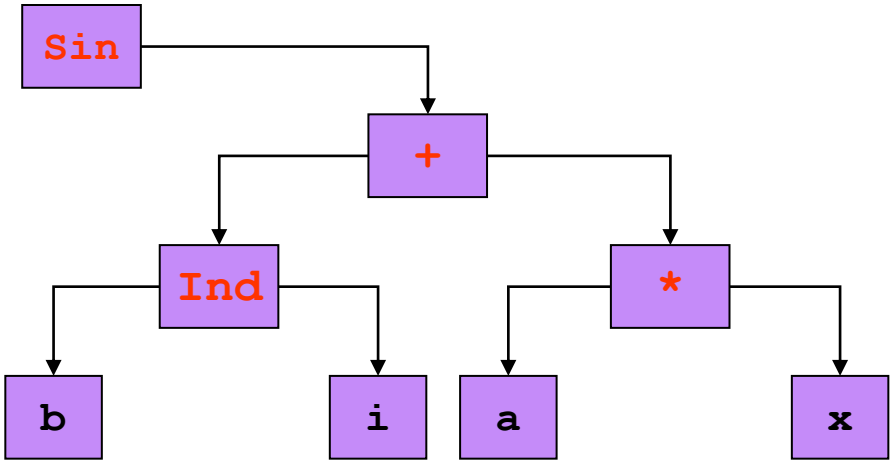
$$\text{BinFa} := \begin{cases} \textit{Üres}\text{Fa} \\ \textit{Rekord}(\text{Elem},\text{BinFa},\text{BinFa}) \end{cases}$$

$$\text{Fa} := \begin{cases} \textit{Üres}\text{Fa} \\ \textit{Rekord}(\text{Elem},\text{Fá}k) \end{cases}$$

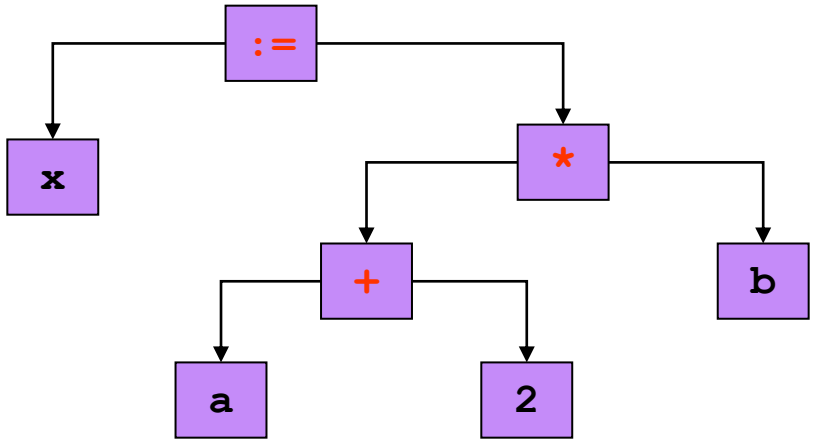


Bináris fa: példák

- `sin(b(i)+a*x)`

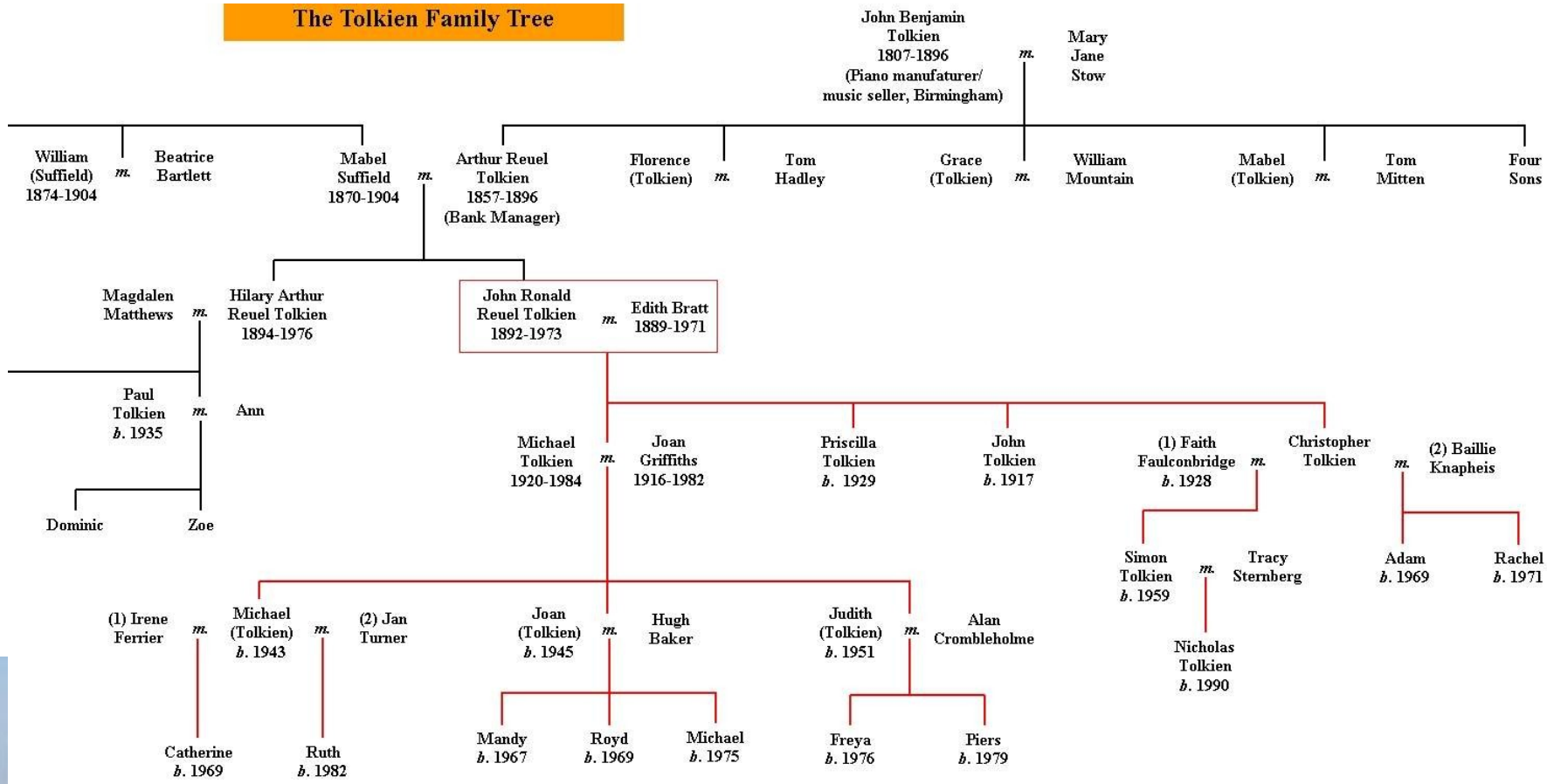


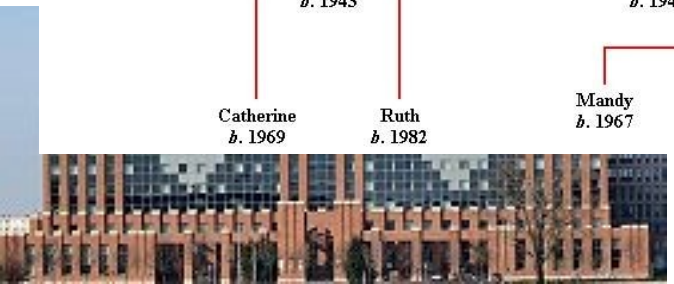
- `x:=(a+2)*b`



Nem bináris fa: példák

John Ronald Reuel Tolkien családfája





Bináris fa

A fa (bináris fa) rekurzív adatszerkezet jellemzői:

- sokaság: azonos típusú elemekből áll;
- akár 0 db elemet tartalmazhat;
- Üres: rekurzív „nullelem”, kitüntetett konstans;
- Fraktál (=önhasonlóság) tulajdonság: a részei ugyanolyan szerkezetűek, mint az egész;
- nem lineárisan rendezett (azaz nem sorozatféle): bármely elemének 0, 1, 2… (közvetlen) rákövetkezője lehet;

- minden elemnek legfeljebb egy (közvetlen) előzője van.

Bináris fa

A fa (bináris fa) rekurzív adatszerkezet ábrázolása:

```
Típus TBinfa=Rekord
          (elem: TElem
           bal,jobb: TBinfa)

Típus TFa=Rekord
          (elem: TElem
           ágak: Sorozat(TFa))
```

A fa elemek sokasága, ezért szükség lesz:

- kezdő- (gyökér) elemre
- aktuális elemre

Bináris fa – dinamikus láncolás

Koncepció:

- a bináris fa egy rekurzív adatszerkezet, ha bármely részét megváltoztatom, az egész meg fog változni;
- megvalósítás dinamikus láncolással;
- a műveletek értékmegosztást feltételeznek.

Megjegyzés: feltesszük, hogy az előfeltételeket mindenhol betartjuk, nincs ellenőrzés.

Bináris fa – dinamikus láncolás

A Bináris fa rekurzív adatszerkezet műveletei:

`Üres:Binfa`

`üres?(Binfa):Logikai`

`egyeleműfa(Elem):Binfa`

`Balrailleszt(Binfa,Binfa):Binfa`

előfeltétel: Binfa nem üres, a bal része üres

`Jobbrailleszt(Binfa,Binfa):Binfa`

előfeltétel: Binfa nem üres, a jobb része üres

`gyökérelem(Binfa):Elem`

előfeltétel: Binfa nem üres

Bináris fa – dinamikus láncolás

A Bináris fa rekurzív adatszerkezet műveletei:

`balgyerek(Binfa):Binfa`

előfeltétel: Binfa nem üres

`jobbgyerek(Binfa):Binfa`

előfeltétel: Binfa nem üres

`gyökérmódosít(Binfa,Elem):Binfa`

előfeltétel: Binfa nem üres

Bináris fa – dinamikus láncolás

A Bináris fa rekurzív adatszerkezet műveletei:

```
Üres(bf):
  bf:=sehova
Eljárás vége.
üres?(bf):
  üres?:=bf=sehova
Függvény vége.
egyeleműfa(Elem):
  Lefoglal(bf,(Elem,sehova,sehova))
  egyeleműfa:=bf
```

```
Függvény vége.
```

Bináris fa – dinamikus láncolás

A Bináris fa rekurzív adatszerkezet műveletei:

```
Balrailleszt(bf,rf):
  Tartalom(bf).bal:=rf
Eljárás vége.

Jobbrailleszt(bf,rf):
  Tartalom(bf).jobb:=rf
Eljárás vége.

gyökérelem(bf):
  gyökérelem:=Tartalom(bf).elem
Függvény vége.
```

Bináris fa – dinamikus láncolás

A Bináris fa rekurzív adatszerkezet műveletei:

```
balgyerek(bf):
  balgyerek:=Tartalom(bf).bal
Függvény vége.
jobbgyerek(bf):
  jobbgyerek:=Tartalom(bf).jobb
Függvény vége.
Gyökérmódosít(bf,Elem):
  Tartalom(bf).elem:=Elem
Eljárás vége.
```

Bináris fa – statikus láncolás

Koncepció:

- a bináris fa egy sokaság, amelyben a sorrend speciális;
- megvalósítás statikus láncolással;
- ha törlést nem engedünk meg, akkor az új elemek csak a tömb végére kerülhetnek;
- a műveletek a struktúrában való mozgást feltételeznek.

```
Típus TBinfa=Rekord
 (gyökér,akt,szabad: 0..Max,
  t: tömb(1..Max,TBinfaelem))

Tbinfaelem=Rekord
 (elem: TElem, bal,jobb: 0..Max)
```

Bináris fa – statikus láncolás

A Bináris fa rekurzív adatszerkezet műveletei:

`Üres:Binfa`

`üres?(Binfa):Logikai`

`Egyeleműfa(Elem):Binfa`

`Balrailleszt(Binfa,Elem):Binfa`

előfeltétel: Binfa aktuális eleme nem üres, a bal része üres

`Jobbrailleszt(Binfa,Elem):Binfa`

előfeltétel: Binfa aktuális eleme nem üres, a jobb része üres

`Gyökérre(Binfa):Binfa`

Bináris fa – statikus láncolás

A Bináris fa rekurzív adatszerkezet műveletei:

`elem(Binfa):Elem`

előfeltétel: Binfa aktuális eleme nem üres

`Balra(Binfa):Binfa`

előfeltétel: Binfa aktuális eleme nem üres

`Jobbra(Binfa):Binfa`

előfeltétel: Binfa aktuális eleme nem üres

`Módosít(Binfa,Elem):Binfa`

előfeltétel: Binfa aktuális eleme nem üres

Bináris fa – statikus láncolás

A Bináris fa rekurzív adatszerkezet műveletei:

```
Üres(bf):
  bf.gyökér:=0; bf.akt:=0; bf.szabad:=1
Eljárás vége.

üres?(bf):
  üres?:=bf.gyökér=0
Függvény vége.

Egyeleműfa(bf,Elem):
  bf.gyökér:=1; bf.akt:=1; bf.t(1):=(Elem,0,0)
Eljárás vége.
```

```
Gyökérre(bf):
  bf.akt:=bf.gyökér
Eljárás vége.
```

Bináris fa – statikus láncolás

A Bináris fa rekurzív adatszerkezet műveletei:

```
Balrailleszt(bf,Elem):
  x:=bf.szabad; bf.akt:=x; bf.t(bf.akt).bal:=x
  bf.t(x):=(Elem,0,0); bf.szabad:=bf.szabad+1
Eljárás vége.
```

```
Jobbrailleszt(bf,Elem):
  x:=bf.szabad; bf.akt:=x; bf.t(bf.akt).jobb:= x
  bf.t(x):=(Elem,0,0); bf.szabad:=bf.szabad+1
Eljárás vége.
```

```
elem(bf):
  elem:=bf.t(bf.akt).elem
Függvény vége.
```

Bináris fa – statikus láncolás

A Bináris fa rekurzív adatszerkezet műveletei:

```
Balra(bf):
  bf.akt:=bf.t(bf.akt).bal
Függvény vége.
Jobbra(bf):
   bf.akt:=bf.t(bf.akt).jobb
Függvény vége.
Módosít(bf,Elem):
  bf.t(bf.akt).elem:=Elem
Eljárás vége.
```

Bináris fa – statikus láncolás

Megjegyzések:

- A statikus láncolással megvalósított bináris fa (aktuális elemmel és gyökérelemmel) dinamikus láncolással is menne.
- A dinamikus láncolással megvalósított bináris fa az értékmegosztás miatt statikus láncolással nem működne.

Bináris fa – statikus láncolás

Módosítás: a bináris fa egyes elemei tartalmazhatják a fölöttük levő elem azonosítóját is.

```
Típus TBinfa=Rekord
   (gyökér,akt: 0..Max,
    t: tömb(1..Max,TBinfaelem))

Tbinfaelem=Rekord
   (elem: TElem,
    bal,jobb,szülő: 0..Max)
```

Bináris fa – dinamikus láncolás

Módosítás: a bináris fa egyes elemei tartalmazhatják a fölöttük levő elem azonosítóját is – dinamikus láncolással.

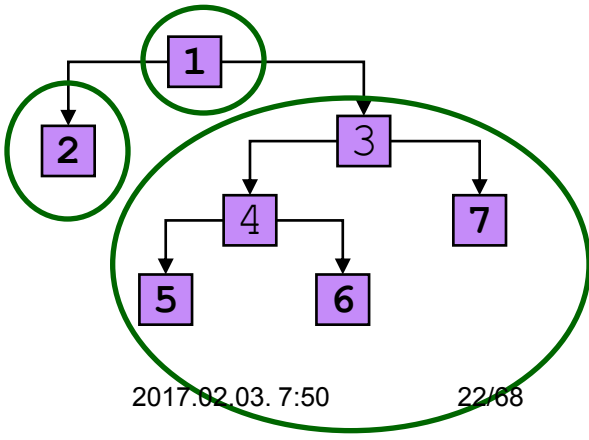
```
Típus TBinfa=Rekord
   (gyökér,akt: TBinfaelem'Mutató)

TBinfaelem=Rekord
   (elem: TElem,
    bal,jobb,szülő: TBinfaelem'Mutató)
```

Bináris fa – új műveletek

Fa bejárások: Bal-közép-jobb

```
BKJ(bf):
    Ha nem üres?(bf) akkor
                BKJ(balgyerek(bf))
                Ki: gyökérelem(bf)
                BKJ(jobbgyerek(bf))
  Elágazás vége
Eljárás vége.
```

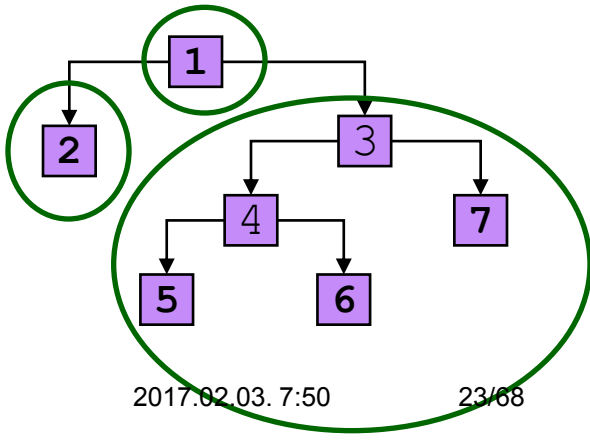
Alkalmazás: rendezőfa



Bináris fa – új műveletek

Fa bejárások: Közép-bal-jobb

```
KBJ(bf):
   Ha nem üres?(bf) akkor
                Ki: gyökérelem(bf)
                KBJ(balgyerek(bf))
                KBJ(jobbgyerek(bf))
   Elágazás vége
Eljárás vége.
```

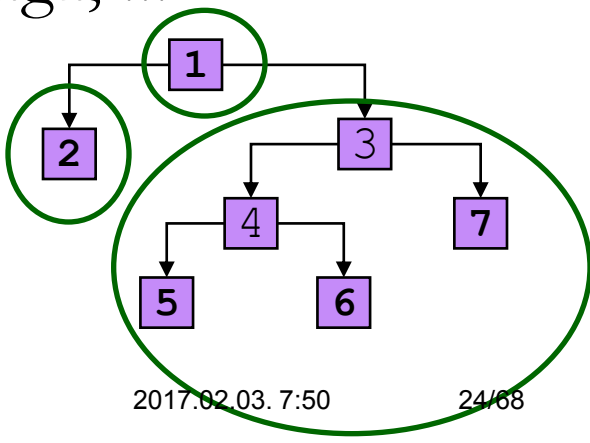


Bináris fa – új műveletek

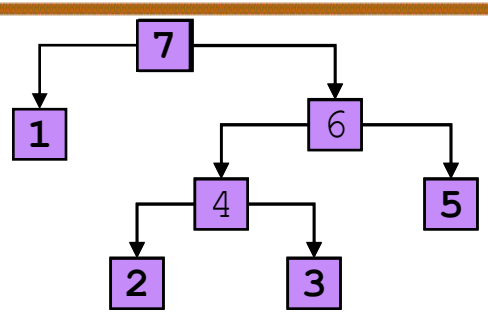
Fa bejárások: Bal-jobb-közép

```
BJK(bf):
   Ha nem üres?(bf) akkor
                BJK(balgyerek(bf))
                BJK(jobbgyerek(bf))
                Ki: gyökérelem(bf)
   Elágazás vége
Eljárás vége.
```

Alkalmazás: Fa törlése, elemszáma, magassága, ...



Bináris fa – új műveletek



Fa törlése (dinamikus láncolással)

```
Törlés(bf):
  Ha nem üres?(bf) akkor
            Törlés(balgyerek(bf))
            Törlés(jobbgyerek(bf))
            Gyökértörlés(bf)
  Elágazás vége
Eljárás vége.
```

```
Gyökértörlés(bf):
  Felszabadít(bf); bf:=sehova
Eljárás vége.
```

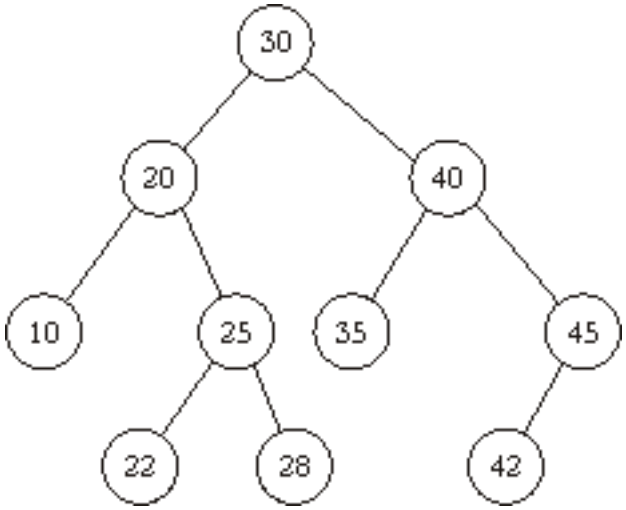
Kereső- és rendezőfák

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.



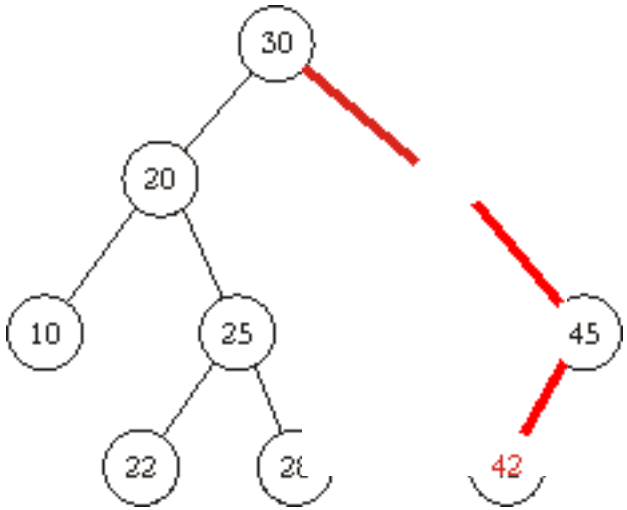
Kereső- és rendezőfák

A keresőfa új műveletei:

- keresés
- beillesztés
- törlés
- kiegyensúlyozás

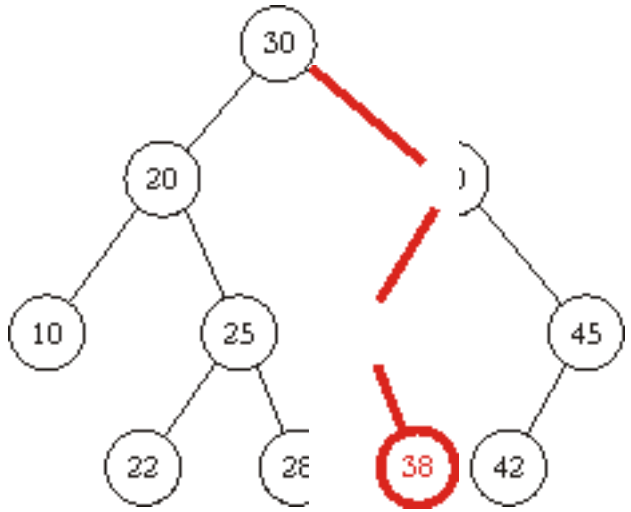
Kereső- és rendezőfák

```
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

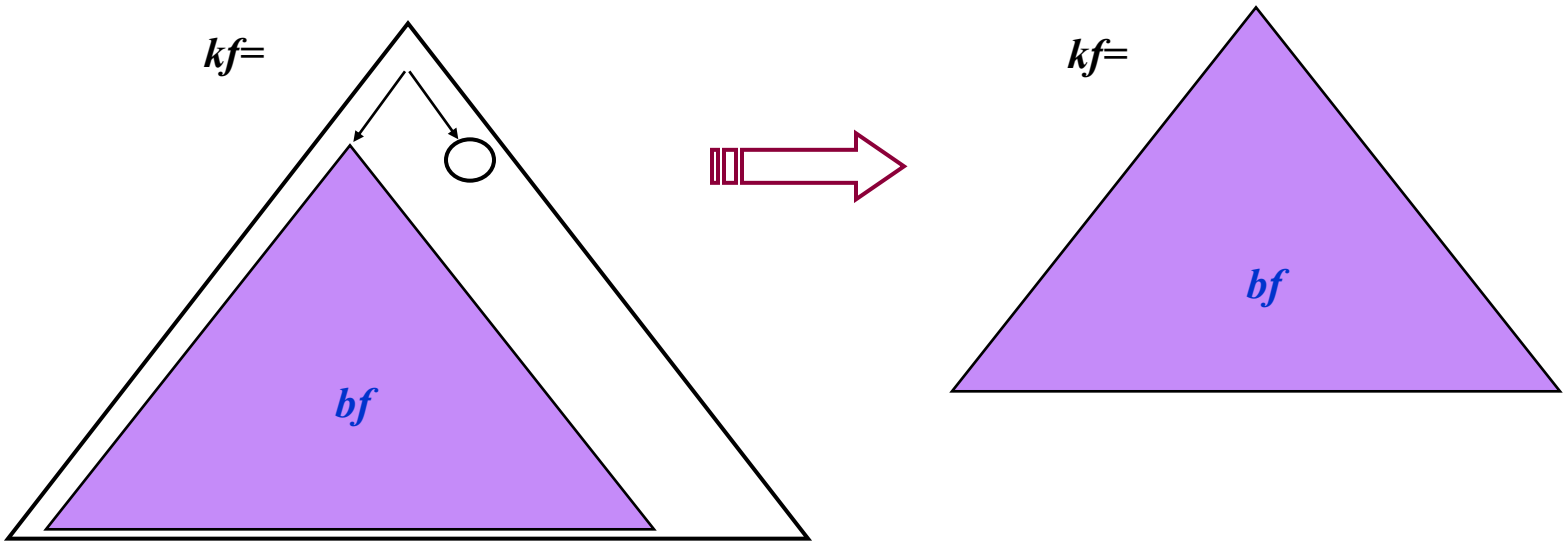
```
Törlés(k,kf):
  Elágazás
    k<gyökérelem(kf).kulcs
       esetén Törlés(k,balgyerek(kf))
    k>gyökérelem(kf).kulcs
       esetén Törlés(k,jobbgyerek(kf))
    k=gyökérelem(kf).kulcs
       esetén Gyökértörlés(k,kf)
  Elágazás vége
Függvény vége.
```

Kereső- és rendezőfák

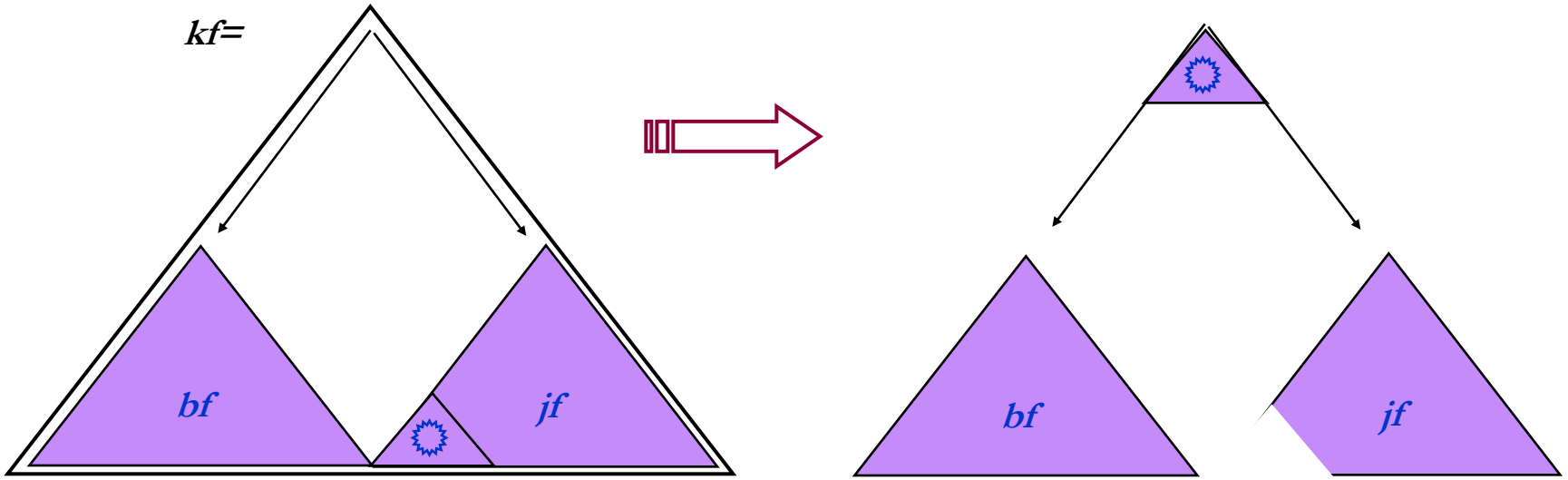
A megoldandó estek: a törlendő

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

Kereső- és rendezőfák



Kereső- és rendezőfák



Kereső- és rendezőfák

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 tejes 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.
```

Fák

Néhány speciális fa művelet:

```
elemszám(bf):
  Ha üres?(bf) akkor elemszám:=0
  különben elemszám:=1+elemszám(balgyerek(bf))+
                      elemszám(jobbgyerek(bf))
Függvény vége.

magas(bf):
  Ha üres?(bf) akkor magas:=0
  különben magas:=1+max(magas(balgyerek(bf)),
                        magas(jobbgyerek(bf))
Függvény vége.
```

Fák

```
szélesség(bf,szint):
  Ha nem üres?(bf)
     akkor sz(szint):=sz(szint)+1
           szélesség(balgyerek(bf),szint+1)
           szélesség(jobbgyerek(bf),szint+1)
Eljárás vége.
széles(bf):
  szélesség(bf,0); max:=0;
  Ciklus i=1-től magas(fb)-ig
    Ha sz(i)>sz(max) akkor max:=i
  Ciklus vége
Eljárás vége.
```

Fák

Bal- és jobbgyerekszám közötti különbségek maximuma:

```
különbség(bf):
  Ha üres?(bf) akkor különbség:=0
  különben
    különbség:=max(abs(elemszám(balgyerek(bf))-
                       elemszám(jobbgyerek(fb)),
                   különbség(balgyerek(bf)),
                   különbség(jobbgyerek(bf))
Függvény vége.
```

Nagyon lassú megoldás a többszörös rekurzió miatt.

Fák

```
különbség(bf,db):
  Ha üres?(bf) akkor különbség:=0; db:=0
  különben a:=különbség(balgyerek(bf),db1)
           b:=különbség(jobbgyerek(bf),db2)
           db:=db1+db2+1
           különbség:=max(abs(db1-db2),a,b)
Függvény vége.
```

Kérdezőfa

A kérdezőfa egy olyan bináris fa, amelyre teljesülnek az alábbi tulajdonságok:

- A fának n levele van, amelyek balról jobbra sorrendben az 1,…,n számokat tartalmazzák.
- A fának n-1 belső pontja van, mindegyiknek 2 gyereke.
- Minden p belső pont a p bal-részfájában levő levélértékek maximumát tartalmazza.
- Minden kérdéshez költségek tartoznak, vagy a kérdések száma korlátozott vagy a nem válaszok száma korlátozott...

A kérdezőfát úgy építjük fel, hogy a kérdések összköltsége minimális legyen!

Kérdezőfa

A kérdezőfa használata:

- Induljunk ki a fa gyökeréből (az 1-es elemből)!
- Amíg az aktuális pont nem levél, kérdezzünk rá az aktuális ponthoz tartozó értékre (a keresett érték ≤-e nála)!
- Ha a válasz igen, akkor lépjünk a fában balra, egyébként pedig jobbra!

Statikus láncolással:

```
Típus Kérdezőfa=Tömb(1..Max,Faelem)
      Faelem=Rekord(érték,ár: Egész,
                    bal,jobb: 0..max)
```

Kérdezőfa

```
Kérdés(kf,p,S):
  p:=1; S:=0
  Ciklus amíg kf(p).bal>0 {vagy kf(p.jobb)>0}
    S:=S+kf(p).érték; Be: V
    Ha V="igen" akkor p:=kf(p).bal
                különben p:=kf(p).jobb
  Ciklus vége
Eljárás vége.
```

Kérdezőfa

Kérdezőfa előállítása kérdésköltségek esetén:

- Jelölje K(x) a kérdezőfában a gyökértől az x levélig haladó úton a belső pontok kérdéseihez értékek összegét.
- Ha a gondolt szám x, akkor a kitalálásának költsége K(x).
- Az a kérdezőfa optimális, amelyre a K(x) értékek maximuma a lehető legkisebb.

A kérdezőfa előállítása:

dinamikus progarmozás!

Nem bináris fák

A fa rekurzív adatszerkezet jellemzői:

- sokaság: azonos típusú elemekből áll;
- akár 0 db elemet tartalmazhat;
- Üres: rekurzív „nullelem”, kitüntetett konstans;
- Fraktál (=önhasonlóság) tulajdonság: a részei ugyanolyan szerkezetűek, mint az egész;
- nem lineárisan rendezett (azaz nem sorozatféle): bármely elemének 0, 1, 2… (közvetlen) rákövetkezője lehet;

- minden elemnek legfeljebb egy (közvetlen) előzője van.

Nem bináris fák

A fa rekurzív adatszerkezet ábrázolása:

```
Típus TFa=Rekord
          (elem: TElem
           ágak: Sorozat(TFa))
```

Egy speciális változat (ágszám felső korláttal):

```
Típus TFa=Rekord
          (elem: TElem
           db: Egész
           ág: Tömb(1..Max,TFa))
```

Nem bináris fák

A fa rekurzív adatszerkezet műveletei:

```
Üres:Fa
üres?(Fa):Logikai
Egyeleműfa(Elem):Fa
gyerekszám(Fa):Egész
Beilleszt(Fa,Fa):Fa
Gyökérelem(Fa):Elem
Gyökérmódosít(Fa,Elem):Fa
Gyerek(Fa,Egész):Fa
```

Nem bináris fák

A fa műveletei megvalósítása dinamikus láncolással:

```
Üres(f):
  f:=sehova
Eljárás vége.
üres?(f):
  üres?:=(f=sehova)
Függvény vége.
Egyeleműfa(e):
  Lefoglal(f,(e,Üressorozat)); Egyeleműfa:=f
Függvény vége.
```

Nem bináris fák

A fa műveletei megvalósítása dinamikus láncolással:

```
gyerekszám(f):
  gyerekszám:=Elemszám(Tartalom(f).ágak)
Függvény vége.
Gyökérelem(f):
  Gyökérelem:=Tartalom(f).elem
Függvény vége.
Gyökérmódosít(f,e):
  Tartalom(f).elem:=e
Eljárás vége.
```

```
gyerekszám(f):
  gyerekszám:=Tartalom(f).db
Függvény vége.
```

Nem bináris fák

Sorozattal ábrázolva:

```
Beilleszt(mire,mit):
  Tartalom(mire).ágak:=
                  Végére(Tartalom(mire).ágak,mit)
Eljárás vége.
Gyerek(f,i):
  Ha i≤Elemszám(Tartalom(f).ágak)
     akkor Gyerek:=Tartalom(f).ágak(i)
  különben Gyerek:=Üres
Függvény vége.
```

Nem bináris fák

Tömbbel ábrázolva:

```
Beilleszt(mire,mit):
  Tartalom(mire).db:=Tartalom(mire).db+1
  Tartalom(mire).ág(Tartalom(mire).db):=mit
Eljárás vége.

Gyerek(f,i):
  Ha i≤Tartalom(f).db
     akkor Gyerek:=Tartalom(f).ág(i)
  különben Gyerek:=Üres
Függvény vége.
```

Nem bináris fák – alkalmazás

A fa elemszáma

```
elemszám(f):
  Ha üres?(f) akkor elemszám:=0
  különben s:=1
           Ciklus i=1-től Tartalom(f).db-ig
             s:=s+elemszám(Tartalom(f).ág(i))
           Ciklus vége
           elemszám:=s
Függvény vége.
```

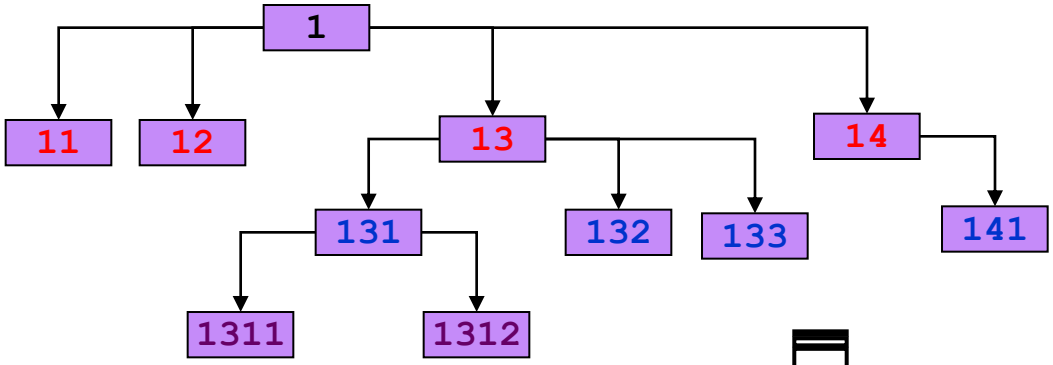
Nem bináris fák – alkalmazás

A fa magassága

```
magasság(f):
   Ha üres?(f) akkor magasság:=0
   különben max:=0
            Ciklus i=1-től Tartalom(f).db-ig
              m:=magasság(Tartalom(f).ág(i))
              Ha m>max akkor max:=m
            Ciklus vége
            magasság:=max+1
Függvény vége.
```

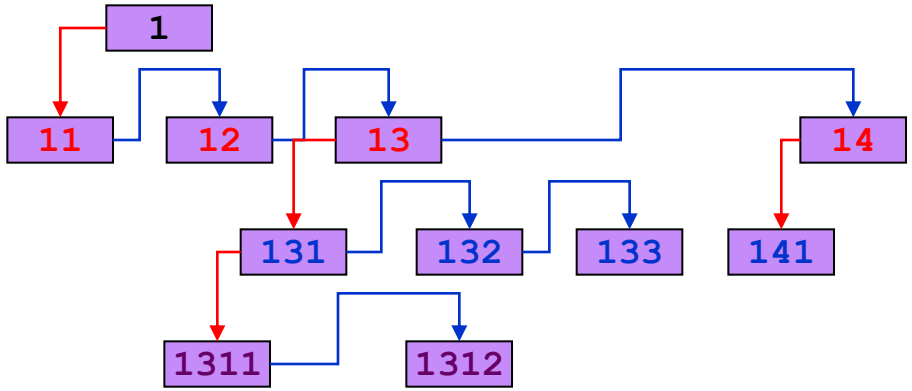
Nem bináris fák – binárisan

Bináris ábrázolás:



balra az 1. gyerek

jobbra a következő testvér



Nem bináris fák – binárisan

```
Gyerek(f,i):
  Ha i=1 akkor Gyerek:=Tartalom(f).bal
  különben Gyerek:=Testvér(Tartalom(f).bal,i)
Függvény vége.
Testvér(f,i):
  Ha i=2 akkor Testvér:=Tartalom(f).jobb
  különben Testvér:=Testvér(Tartalom(f).jobb,i-1)
Függvény vége.
```

Nem bináris fák – binárisan

```
Elsőgyerek(f):
  Elsőgyerek:=Tartalom(f).bal
Függvény vége.
Következőgyerek(gy):
  Következőgyerek:=Tartalom(gy).jobb
Függvény vége.
Vanméggyerek(gy):
  Vanméggyerek:=(Tartalom(gy).jobb≠sehova)
Függvény vége.
```

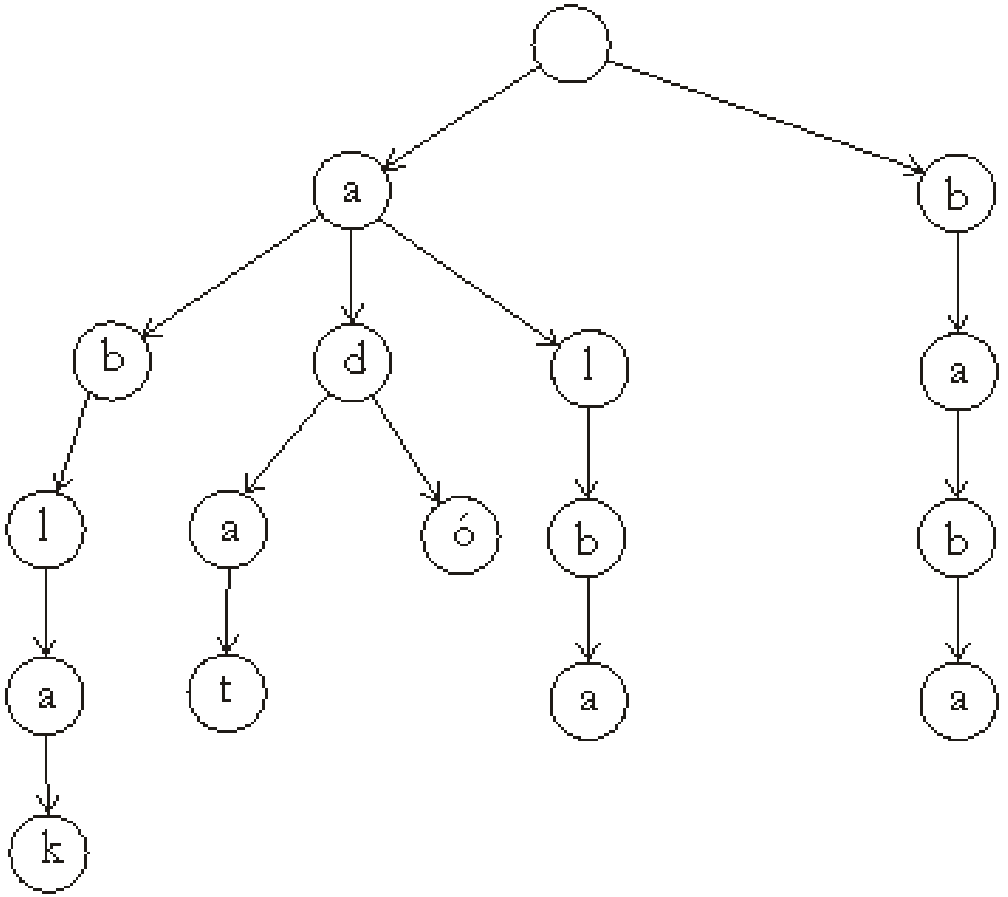
Nem bináris fák – binárisan

```
Gyerek(f,i):
  gy:=Elsőgyerek(f)
  Ha i=1 akkor Gyerek:=gy
       különben Ciklus j=2-től i-ig
                  gy:=Következőgyerek(gy)
                Ciklus vége
                Gyerek:=gy
Függvény vége.
```

Szófák

A szófa:

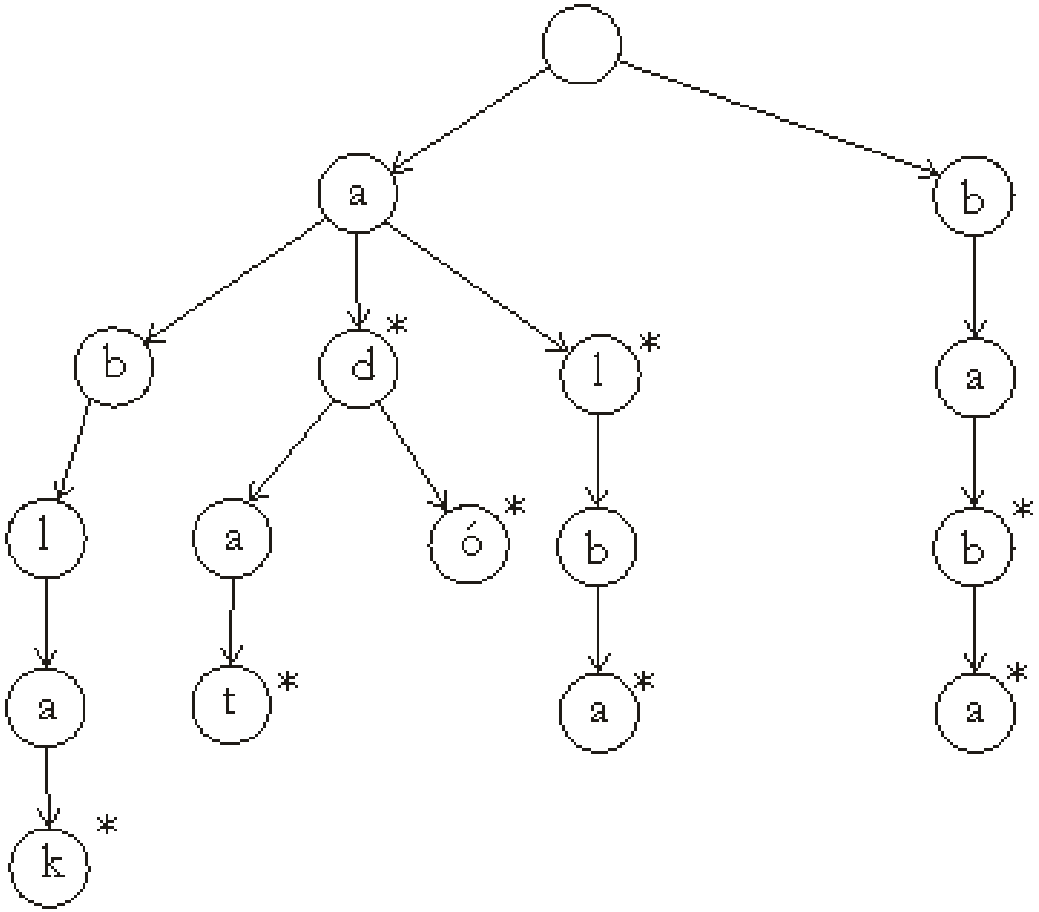
- nem bináris fa
- a gyökere üres
- a szavak felülről lefelé olvashatók
- hol a szó vége?
- hány gyerek lehet?



Szófák

A szófa:

- jelöljük a szóvégeket!



Szófák

A szófa ábrázolása:

- indexeljük a következő betűvel a szófa további részét;
- jelezzük, hogy szó végén vagyunk-e!

```
Típus SzóFa=Rekord
                (betű:Tömb('a'..'z':Szófa,
                vége:Logikai)
```

Szófák

Szó keresése:

```
Bennevan?(szf,s):
  Ha Tartalom(szf).vége és üres?(s)
     akkor Bennevan?:=igaz
  különben ha üres?(s) akkor Bennevan?:=hamis
  különben ha üres?(Tartalom(szf).betű(első(s)))
     akkor Bennevan?:=hamis
  különben Bennevan?:=Bennevan?(Tartalom(szf).
                    betű(első(s)),elsőnélküli(s))
Függvény vége.
```

Szófák

Szó beillesztése:

```
Beilleszt(szf,s):
  Ha üres?(s) akkor Tartalom(szf).vége:=igaz
  különben
    Ha üres?(Tartalom(szf).betű(első(s)))
       akkor Lefoglal(Tartalom(szf).betű(első(s))
    Beilleszt(Tartalom(szf).betű(első(s)),
                elsőnélküli(s))
Függvény vége.
```

Szófák

Problémák:

- minden elemhez annyi mutató kell, ahány betű van az ábécében;
- kérdéses a magyar ábécével való indexelés;
- sok faelemnél nincs is elágazás, azaz egyetlen ág meg tovább.

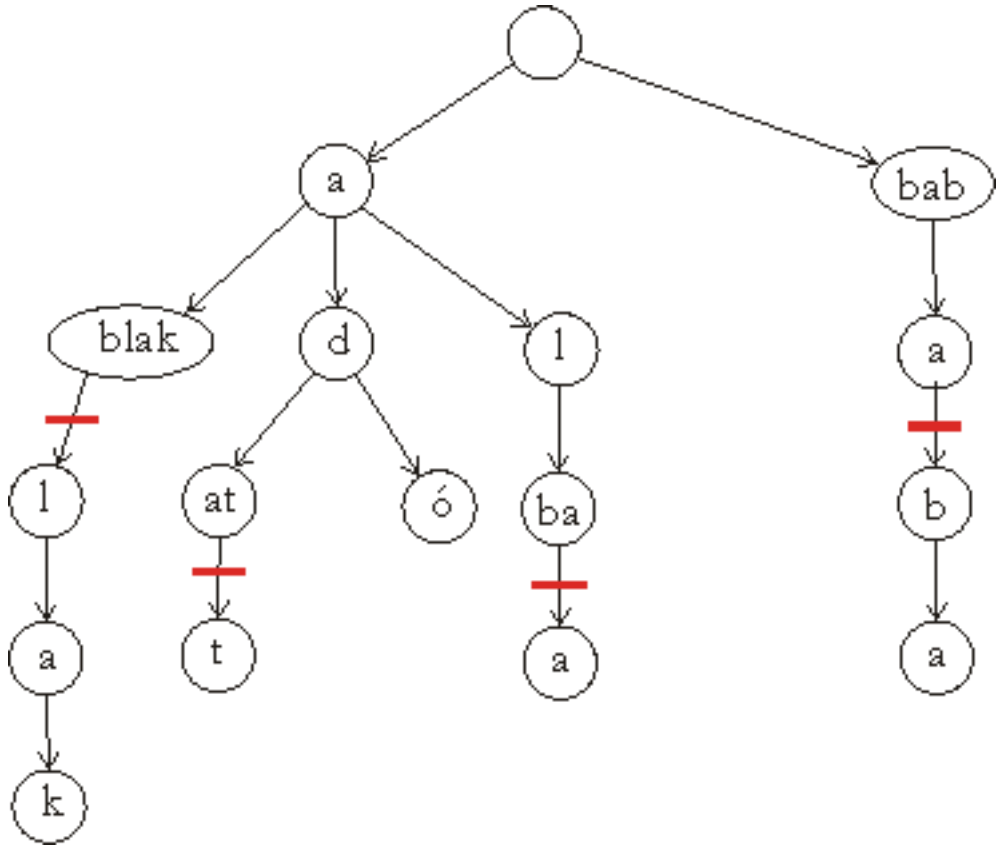
Ötlet: Indexelés helyett logaritmikus keresés rendezett tömbben.

Kérdés: Lineáris is elég rendezetlen tömbben?

Szófák

A szófa tömörítése:

- vonjuk össze az elágazás nélküli csomópontokat!



Fa megadása sorozattal

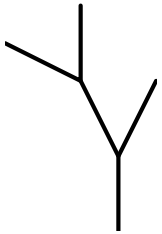
Egy fát zárójelekkel és F betűkkel írunk le : (baloldali ág) F (jobboldali ág) formában. A fának törzse biztosan van.

Példa:

ágnélküli fa: F

kétágú fa: (F) F (F)

bonyolultabb fa: ((F) F (F)) F (F)



Fa megadása sorozattal

Írj programot, amely meghatározza

A. a fa magasságát (a leghosszabb út hosszát a gyökértől valamelyik ág végéig);

B. a fa tömegét (feltételezve, hogy a törzs, illetve a vele azonos tömegű ágdarabok egységnyi tömegűek);

C. a fa elágazásainak számát!

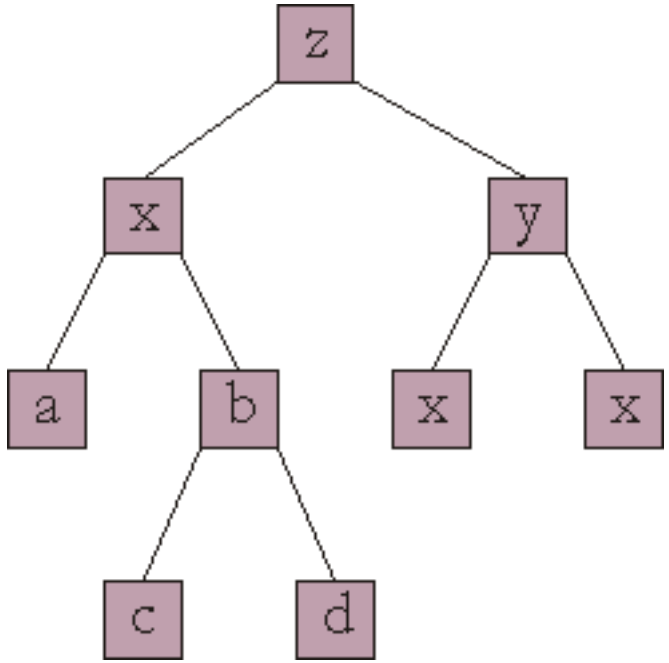
Bináris fa – felépítése

Bináris fák szövegesen is megadhatók az alábbi szabályok szerint is:

1) Minden karakter, ami az angol ábécé eleme, faleírás.

2) Ha x karakter valamint f1 és f2 faleírás, akkor az x(f1,f2) szöveg is faleírás.

Pl.: z(x(a,b(c,d)),y(x,x))



Bináris fa – felépítése

```
Felépít(bf,s,i):
  Lefoglal(bf,(s[i],sehova,sehova)); i:=i+1
  Ha i≤hossz(s) akkor
    Ha s[i]='(' akkor
       i:=i+1; Felépít(Tartalom(bf).bal,s,i)
    különben ha s[i]=',' akkor
       i:=i+1; Felépít(Tartalom(bf).jobb,s,i)
    különben {')'} i:=i+1
Eljárás vége.
```

Fák, bináris fák