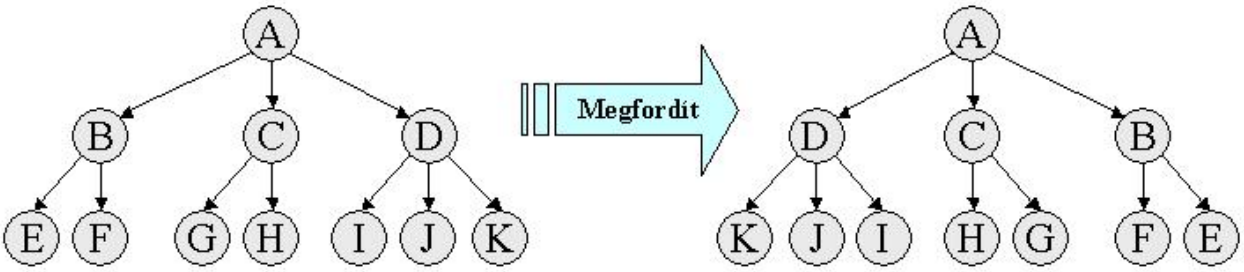
A

1a. feladat: **10**

*Adott az alábbi rekurzív típus. Írja meg a Megfordít függvény **rekurzív algoritmus**át a „nagy a kicsiben” elv értelmezése szerint! (Azaz a használható rekurzió-műveletek:* `Üres?`, `Üres`, `Mező`, `Legyen`.*)*

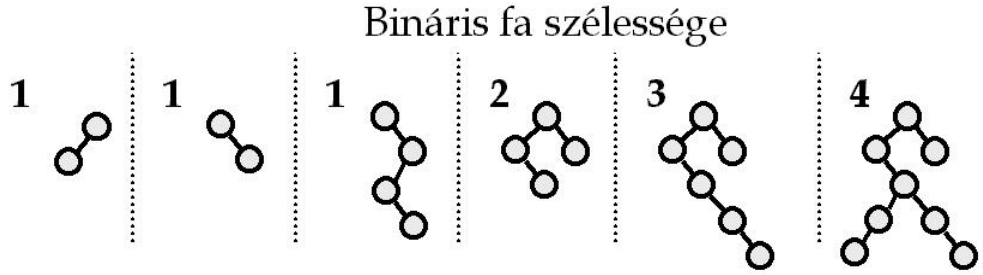
```
Típus TTriFa=Rekurzió(elem:TElem,bal,közép,jobb:TTriFa)
```



2a. feladat: **5+10**

*Írjon **rekurzív specifikáció**t (5) és **iteratív algoritmus**t (10) a következő BinFa-művelethez:*

```
Függvény Szélesség(Konstans f:BinFa):Egész
[Uf: a „balra leginkább kilógó” és a „jobbra leginkább kilógó” elem
    „távolsága”, azaz a gyökértől balra, ill. jobbra megtehető lépések
    számának különbségének abszolút értéke. L. a magyarázó ábrát!]
```



3a. feladat: **5+5**

*Két pozitív egész szám **szorzás**a (A*B) visszavezethető összeadásra és eggyel való csökkentésre, amelyek műveletigénye: Θ(log(A*B))[*], illetve Θ(logA). Így a „bamba” algoritmus Θ(A*(Θ(log(A*B)) + Θ(logA))) vagyis Θ(A*log(A*B)) műveletigényű. A szorzást hatékonyabban is elvégezhetjük a 2-vel való szorzásra / osztásra visszavezetve a következőképpen: ha B páros, akkor számoljuk ki a (2*A) és (B/2) számok szorzatát, ha pedig páratlan, akkor az A+A*(B-1)-et. Készítse el az ennek megfelelő **rekurzív** szorzó **algoritmus**t! (5) Vegyük észre: a 2-vel való szorzás, ill. osztás nem más, mint 1-bittel való léptetések, így olcsó műveletek! Mekkora a módosított algoritmusának műveletigénye? Adja meg a „levezetésének” gondolatmenetét! (5)*

4a. feladat: **10**

*Adott egy **bináris fa**. Készítsük el olyan **másolat**át, amely elemtípusa egy „hossz” mezővel egészül ki, és minden elem ezen mezőjébe a gyökértől vett „távolsága” (az odáig szükséges lépések száma) kerül. Egyébként a fa változatlan.*

Tervezett ponthatárok:

Kettes, ha legalább:	*14*	*Négyes, ha legalább:*	*30*
Hármas, ha legalább:	*22*	*Ötös, ha legalább:*	*38*
		Maximum:	*10+15+10+10=45*

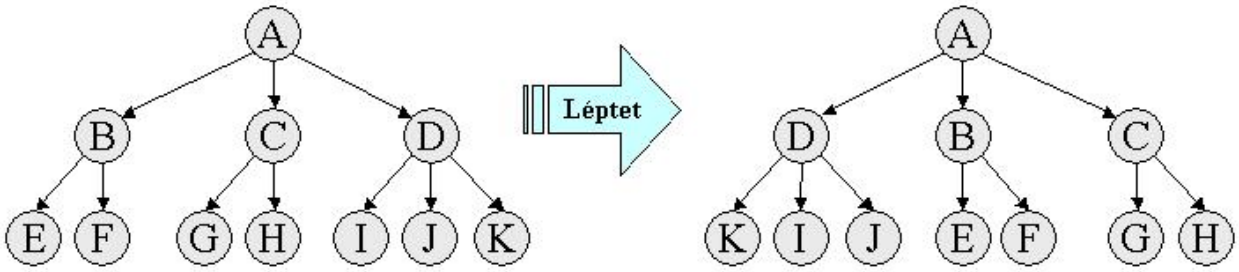
[*] Hiszen a X érték bináris ábrázolásához *logX* bitre van szükség, és föltehetjük, hogy két ilyen nagyságrendű szám összeadásához (eggyel csökkentéséhez) éppen bit-hosszúságú lépésre van szükség.

$\mathcal{B}$

1b. feladat: **10**

*Adott az alábbi rekurzív típus. Írjuk meg a Léptet eljárás **rekurzív algoritmus**át a „nagy a kicsiben” elv értelmezése szerint! (Azaz a használható rekurzió-műveletek:* `Üres?`, `Üres`, `Mező`, `Legyen`.*)*

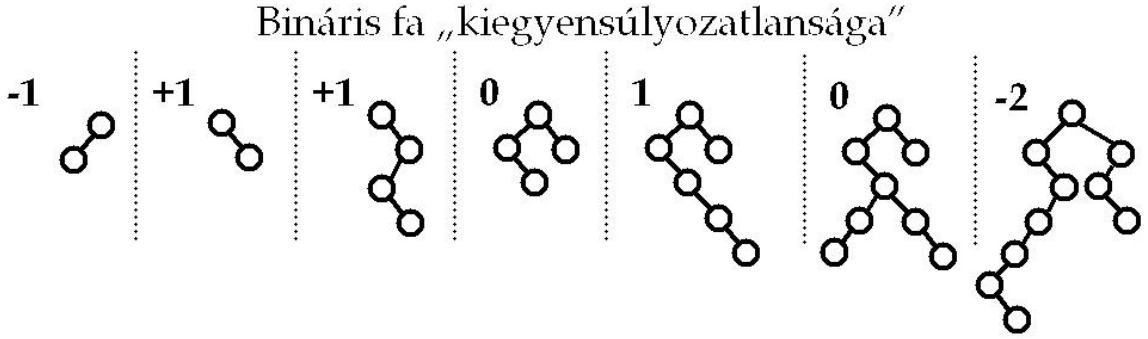
```
Típus TTriFa=Rekurzió(elem:TElem,bal,közép,jobb:TTriFa)
```



2b. feladat: **5+10**

*Írjon **rekurzív specifikáció**t (5) és **iteratív algoritmus**t (10) a következő BinFa-művelethez:*

```
Függvény Kiegysúlyozatlanság(Konstans f:BinFa):Egész
[Uf: a bal és a jobb ágak „szélességének” előjeles eltérése, ahol
     egy fa „bal-szélessége”: a balra leginkább kilógó elem „távolsága” a
     gyökértől, a „jobb-szélesség” hasonló… L. a magyarázó ábrát!]
```



3b. feladat: **5+5**

*Két pozitív egész szám **hatványozás**a (A^B) visszavezethető önmagával való szorzásra és eggyel való csökkentésre, amelyek műveletigénye (B«A)[♥]: $\Theta((\log(A^B))^2)=\Theta(B^2*\log(A)^2)$, illetve $\Theta(\log B)$ [♦]. Így a „bamba” algoritmus $\Theta(B*(\Theta(B^2*\log(A)^2)+\Theta(\log B)))$ vagyis $\Theta(B^3*(\log(A))^2)$ műveletigényű. A hatványozást hatékonyabban is elvégezhetjük a négyzetre emelésre és a 2-vel való osztásra visszavezetve a következőképpen: ha B páros, akkor számoljuk ki a (A*A)^(B/2) hatványt, ha pedig páratlan, akkor az A*A^(B-1)-et. Készítse el az ennek megfelelő **rekurzív** hatványozó **algoritmus**t! (5) Vegyük észre: a 2-vel való osztás 1-bittel való jobbra léptetés művelete, így olcsó művelet! **Mekkora** a módosított algoritmusának **műveletigénye**? Adja meg a „levezetésének” gondolatmenetét! (5)*

4b. feladat: **10**

*Adott egy **bináris fa**. Készítsük el olyan **másolat**át, amely elemtípusa egy „hossz” mezővel egészül ki, és minden elem ezen mezőjébe a gyökértől vett „távolsága” (az odáig szükséges lépések száma) kerül. Egyébként a fa változatlan.*

Tervezett ponthatárok:

Kettes, ha legalább:	*14*	*Négyes, ha legalább:*	*30*
Hármas, ha legalább:	*22*	*Ötös, ha legalább:*	*38*
	Maximum:	*10+15+10+10=45*	

[♥] „«” = „sokkal kisebb”

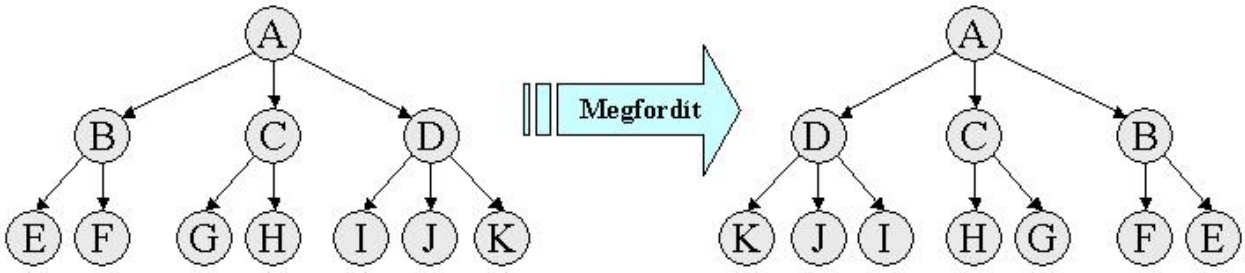
[♦] Hiszen egy X érték bináris ábrázolásához *logX* bitre van szükség, és föltehetjük, hogy két ilyen nagyságrendű szám összeszorzásához $O((\log X)^2)$, eggyel csökkentéséhez $O(\log X)$ lépésre van szükség.

Megoldás

1a. feladat: **10**

*Adott az alábbi rekurzív típus. Írja meg a Megfordít függvény **rekurzív algoritmus**át a „nagy a kicsiben” elv értelmezése szerint! (Azaz a használható rekurzió-műveletek:* `Üres?`, `Üres`, `Mező`, `Legyen`.*)*

```
Típus TTriFa=Rekurzió(elem:TElem,bal,közép,jobb:TTriFa)
```



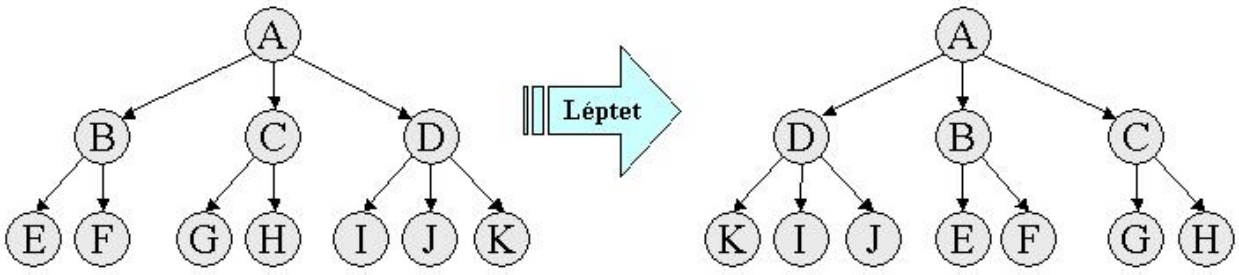
Megoldás:

```
Eljárás Megfordít(Változó f:TTriFa):
  Változó sf:TTRiFa
  Ha nem Üres?(f) akkor
    sf:=f.bal; f.bal:=f.jobb; f.jobb:=sf
    Megfordít(f.bal); Megfordít(f.közép); Megfordít(f.jobb)
  Elágazás vége
Függvény vége.
```

1b. feladat: **10**

*Adott az alábbi rekurzív típus. Írjuk meg a Léptet eljárás **rekurzív algoritmus**át a „nagy a kicsiben” elv értelmezése szerint! (Azaz a használható rekurzió-műveletek:* `Üres?`, `Üres`, `Mező`, `Legyen`.*)*

```
Típus TTriFa=Rekurzió(elem:TElem,bal,közép,jobb:TTriFa)
```



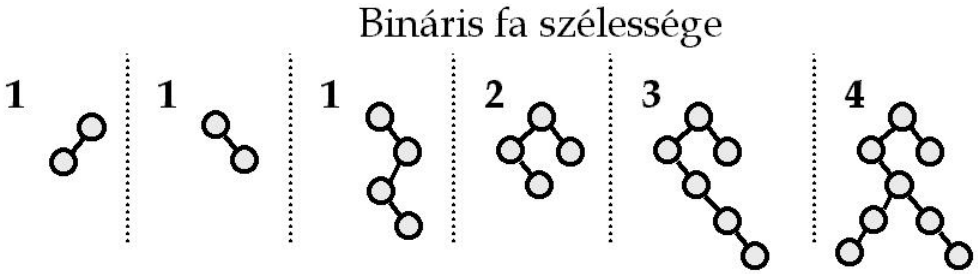
Megoldás:

```
Eljárás Léptet(Változó f:TTriFa):
  Változó sf:TTRiFa
  Ha nem Üres?(f) akkor
    sf:=f.jobb; f.jobb:=f.közép; f.közep:=f.bal; f.bal:=f.sf
    Léptet(f.bal); Léptet(f.közép); Léptet(f.jobb)
  Elágazás vége
Függvény vége.
```

2a. feladat: **5+10**

Írjon rekurzív specifikációt és iteratív algoritmust a következő BinFa-művelethez:

```
Függvény Szélesség(Konstans f:BinFa):Egész
[Uf: a „balra leginkább kilógó” és a „jobbra leginkább kilógó” elem
    „távolsága”, azaz a gyökértől balra, ill. jobbra megtehető lépések
    számának különbségének abszolút értéke. L. a magyarázó ábrát!]
```



Megoldás:

Az Uf-beli rekurzív definíció:

Szélesség: BinFa → Egész

$$\text{Szélesség(bf)} := \begin{cases} 0 & \text{, ha Üres?(bf)} \\ \text{BalSzélesség(BalGyerek(bf))} + \text{JobbSzélesség(JobbGyerek(bf))} & \text{, egyébként} \end{cases}$$

BalSzélesség: BinFa → Egész

$$\text{BalSzélesség(bf)} := \begin{cases} 0 & \text{, ha Üres?(bf)} \\ \text{Max(0,BalSzélesség(JobbGyerek(bf))-1)} & \text{, ha Üres?(BalGyerek(bf))} \\ \text{Max(BalSzélesség(BalGyerek(bf))+1, BalSzélesség(JobbGyerek(bf))-1)} & \text{, egyébként} \end{cases}$$

JobbSzélesség: BinFa → Egész

$$\text{JobbSzélesség(bf)} := \begin{cases} 0 & \text{, ha Üres?(bf)} \\ \text{Max(0,JobbSzélesség(BalGyerek(bf))-1)} & \text{, ha Üres?(JobbGyerek(bf))} \\ \text{Max(JobbSzélesség(JobbGyerek(bf))+1 JobbSzélesség(BalGyerek(bf))-1)} & \text{, egyébként} \end{cases}$$

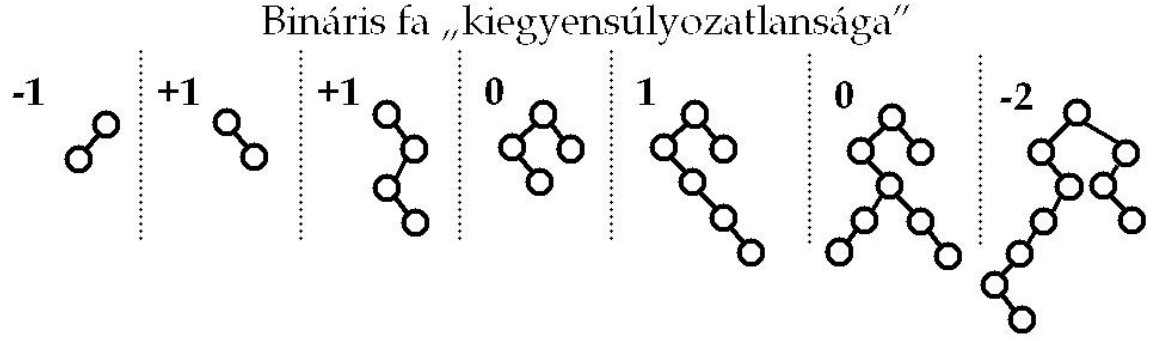
E megoldás „működése” kipróbálható.

Az algoritmus az ismert, iteratív BKJ-bejáráson alapul, kiegészítve azzal, hogy számláljuk a haladás során a balra és jobbra lépéseket (balDb/jobbDb), továbbá feljegyezzük ezek addigi maximumát (balMax/jobbMax). A végén vesszük a két maximum összegét (balMax+jobbMax).

2b. feladat: **5+10**

Írjon rekurzív specifikációt és iteratív algoritmust a következő BinFa-művelethez:

```
Függvény Kiegysúlyozatlanság(Konstans f:BinFa):Egész
[Uf: a bal és a jobb ágak „szélességének” előjeles eltérése, ahol
     egy fa „bal-szélessége”: a balra leginkább kilógó elem „távolsága” a
     gyökértől, a „jobb-szélesség” hasonló… L. a magyarázó ábrát!]
```



Megoldás:

Az Uf-beli rekurzív definíció:

$$\text{Kiegyensúlyozatlanság: BinFa} \rightarrow \text{Egész}$$

$$\text{Kiegyensúlyozatlanság(bf)} := \begin{cases} 0 & \text{, ha Üres?(bf)} \\ -\text{BalSzélesség(BalGyerek(bf))} + \text{JobbSzélesség(JobbGyerek(bf))} & \text{, egyébként} \end{cases}$$

A BalSzélesség és a JobbSzélesség függvényeket l. a 2a magoldásában.

E megoldás „működése” kipróbálható.

Az algoritmus az ismert, iteratív BKJ-bejáráson alapul, kiegészítve azzal, hogy számláljuk a haladás során a balra és jobbra lépéseket (balDb/jobbDb), továbbá feljegyezzük ezek addigi maximumát (balMax/jobbMax). A végén vesszük a két maximum különbségét (-balMax+jobbMax).

3a. feladat: **10**

*Két pozitív egész szám **szorzás**a (A*B) visszavezethető összeadásra és eggyel való csökkentésre, amelyek műveletigénye: $\Theta(\log(A*B))$[♣], illetve $\Theta(\log A)$. Így a „bamba” algoritmus $\Theta(A*(\Theta(\log(A*B)) + \Theta(\log A)))$ vagyis $\Theta(A*\log(A*B))$ műveletigényű. A szorzást hatékonyabban is elvégezhetjük a 2-vel való szorzásra / osztásra visszavezetve a következőképpen: ha B páros, akkor számoljuk ki a (2*A) és (B/2) számok szorzatát, ha pedig páratlan, akkor az A+A*(B-1)-et. Készítse el az ennek megfelelő **rekurzív** szorzó **algoritmus**t! (5) Vegyük észre: a 2-vel való szorzás, ill. osztás nem más, mint 1-bittel való léptetések, így olcsó műveletek! Mekkora a módosított algoritmusának műveletigénye? Adja meg a „levezetésének” gondolatmenetét! (5)*

Megoldás:

A „bamba” megoldás (ez nem volt kérdés!):

$$\text{Szorzás(A,B)} := \begin{cases} 0 & \text{, ha A=0} \\ \text{A+Szorzás(A,EggyelCsökkentés(B))} & \text{, egyébként} \end{cases}$$

A kérdéses hatékonyabb algoritmus:

$$\text{Szorzás(A,B)} := \begin{cases} 0 & \text{, ha A=0} \vee \text{B=0} \\ \text{Szorzás(BalraLéptetés(A),JobbraLéptetés(B))} & \text{, ha 2}\mid\text{B} \\ \text{A+Szorzás(A,EggyelCsökkentés(B))} & \text{, egyébként} \end{cases}$$

Előzetes megjegyzés a hatékonysági számítás elé:

*Ha egy számítás lépésszáma az X szám-paraméter **bit**jeinek a **szám**ától függ, akkor ezt **logX**-szel fejezhetjük ki.*

1. *Páratlan esetben az ismétlés egyetlen lépése: összeadás – $\Theta(\log(A*B))$; EggyelCsökkentés – $\Theta(\log B)$; összesen: $\Theta(\log(A*B))$;*
2. *Páros esetben az ismétlés egyetlen lépése: BalraLéptetés – $\Theta(\log(A*B))$; JobbraLéptetés – $\Theta(\log B)$; összesen: $\Theta(\log(A*B))+\Theta(\log B) = \Theta(\log(A*B))$*

*Mivel a legrosszabb esetben a páros után páratlan eset következik, továbbá a páros esetben a B bitszáma eggyel csökken, ezért a ciklus-lépésszám maximuma 2*logB, a páros-páratlan ciklusmagok „összevont” számítása esetén: az össz-műveletigény: $\Theta(\Theta(\log B)*(\Theta(\log(A*B))+\Theta(\log(A*B)))) = \Theta(\mathbf{log}B*\log(A*B))$*

[♣] Hiszen a X érték bináris ábrázolásához *logX* bitre van szükség, és föltehetjük, hogy két ilyen nagyságrendű szám összeadásához (eggyel csökkentéséhez) éppen bit-hosszúságú lépésre van szükség.

3b. feladat: **5+5**

*Két pozitív egész szám **hatványozás**a (A^B) visszavezethető önmagával való szorzásra és eggyel való csökkentésre, amelyek műveletigénye (B«A)♥: $\Theta((\log(A^B))^2)=\Theta(B^2*\log(A)^2)$, illetve $\Theta(\log B)$ ♦. Így a „bamba" algoritmus $\Theta(B*(\Theta(B^2*\log(A)^2)+\Theta(\log B)))$ vagyis $\Theta(B^3*(\log(A))^2)$ műveletigényű. A hatványozást hatékonyabban is elvégezhetjük a négyzetre emelésre és a 2-vel való osztásra visszavezetve a következőképpen: ha B páros, akkor számoljuk ki a (A*A)^(B/2) hatványt, ha pedig páratlan, akkor az A*A^(B-1)-et. Készítse el az ennek megfelelő **rekurzív** hatványozó **algoritmus**t! (5) Vegyük észre: a 2-vel való osztás 1-bittel való jobbra léptetés művelete, így olcsó művelet! **Mekkora** a módosított algoritmusának **műveletigénye**? Adja meg a „levezetésének" gondolatmenetét! (5)*

Megoldás:

A „bamba" megoldás (ez nem volt kérdés!):

$$\text{Havány}(A,B):=\begin{cases}1 & \text{, ha } B=0\\ A*\text{Hatvány}(A,\text{EggyelCsökkentés}(B)) & \text{, egyébként}\end{cases}$$

A kérdéses hatékonyabb algoritmus:

$$\text{Havány}(A,B):=\begin{cases}\text{Hatvány}(\text{Négyzet}(A),\text{JobbraLéptet}(B)) & \text{, ha } 2\,|\,B\\ A*\text{Hatvány}(A,\text{EggyelCsökkent}(B)) & \text{, egyébként}\end{cases}$$

Hatékonysági számítás:

... az előzőhöz hasonlóan ...

4a. feladat: **10**

*Adott egy **bináris fa**. Készítsük el olyan másolatát, amely elemtípusa egy „hossz" mezővel egészül ki, és minden elem ezen mezőjébe a gyökértől vett „távolsága" (az odáig szükséges lépések száma) kerül. Egyébként a fa változatlan.*

Megoldás:

A bejárás valamelyik algoritmusára építhetünk (legegyszerűbb a KBJ-ra). Amikor egy elem feldolgozásánál tartunk, akkor lemásoljuk, kiegészítjük, a bejárásnak megfelelően a lépésszámlálót módosítjuk, s ez az lépésszám kerül bele.

Egy megoldás Pascal-kódjának részlete:

```
...
  Type
    TElem=String; {=Record ert:String; hossz:Integer End;
                   reperezentálás: 'ert'-mező + ';' + 'hossz'-mező}
  {$i OBinFa.inc}
...
  Procedure Masolas(Const bf:OBinFa Var bf2:OBinFa);
    Procedure Mas(Const bF:OBinFa; Var bF2:OBinFa; Const h:Integer);
      Var sbF,sjF, {bF segédfái}
          sbf2,sjf2:OBinFa; {bF2 segédfái}
          e:TElem;
          hS:String;
    Begin
      With bf do
      Begin
        If not UresE then
        Begin
          e:=GyokerElem; Str(h:0,hS);
          bF2.EgyelemuFa(e+';'+hS);
```

♥ „«" = „sokkal kisebb"

♦ Hiszen egy X érték bináris ábrázolásához *logX* bitre van szükség, és föltehetjük, hogy két ilyen nagyságrendű szám összeszorzásához $O((\log X)^2)$, eggyel csökkentéséhez $O(\log X)$ lépésre van szükség.

```
            BalFa(sbF);  bF2.BalFa(sbf2);  Mas(sbF,sbf2,h+1);
            bF2.BalraIlleszt(sbf2);
            JobbFa(sjF);  bF2.JobbFa(sjf2);  Mas(sjF,sjf2,h+1);
            bF2.JobbraIlleszt(sjf2);
          End
        End;
      End;
    Begin {Masolas}
      kf.Ures;
      Mas(bf,kf,0);
    End; {Masolas}
```

A teljesebb megoldás kipróbálható.

A példaprogramok megnézhetők.