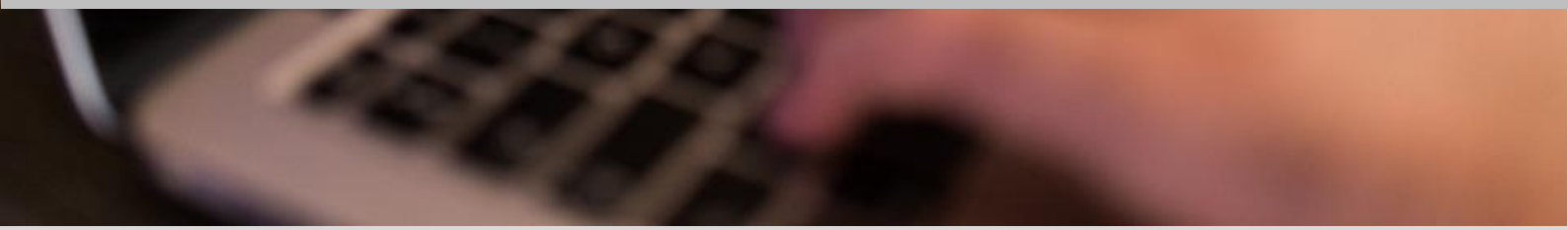
Belépő a tudás közösségébe

Informatika szakköri segédanyag

Rekurzió 2

Bende Imre, Heizlerné Bakonyi Viktória, Menyhárt László, Szlávi Péter, Törley Gábor, Zsakó László

Szerkesztő: Abonyi-Tóth Andor, Zsakó László



A kiadvány „A felsőoktatásba bekerülést elősegítő készségfejlesztő és kommunikációs programok megvalósítása, valamint az MTMI szakok népszerűsítése a felsőoktatásban” (EFOP-3.4.4-16-2017-006) című pályázat keretében készült 2018-ban.

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

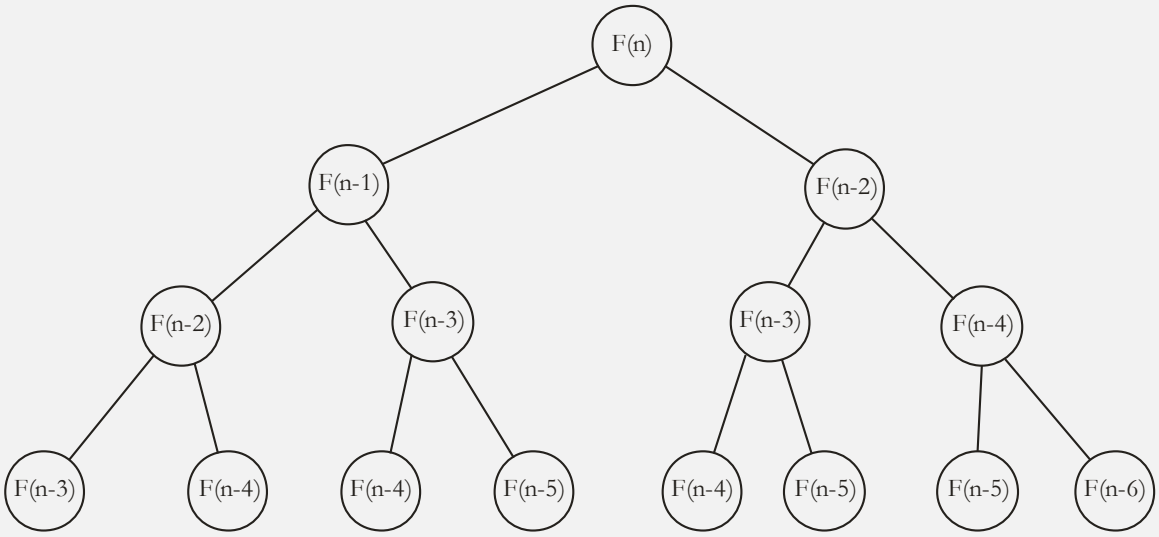
Eötvös Loránd Tudományegyetem
Informatikai Kar

Bajok a rekurzióval

Hely: nagyra dagadt memóriaméret az ismétlődő eljáráshívások miatt.

Idő: a többszörösen ismétlődő hívások felesleges kiszámolása.

Pl. Fibonacci-számoknál:



Az ábráról látható:

- F(N) kiszámolásai száma: 1
- F(N-1) kiszámolásai száma: 1
- F(N-2) kiszámolásai száma: 2
- F(N-3) kiszámolásai száma: 3
- F(N-4) kiszámolásai száma: 5
- ebből következően F(N-5) kiszámolásai száma: 8

Ezek a kiszámolási számok éppen a Fibonacci számok.

Egy kis matematika:

Legyen r(N):=az N. Fibonacci-szám kiszámításához szükséges függvényhívások száma!

$$r(0):=1,\ r(1):=1,\ r(i):=r(i-1)+r(i-2)+1$$

Állítás:

a) $r(i)=F(i+1)+F(i)+F(i-1)-1$, ha $i>1$

b) $r(i)=2*F(i+1)-1$, ahol $F(i)$=az i. Fibonacci-szám.

Megoldási ötlet: amit már kiszámoltunk egyszer, azt ne számoljuk újra! Tároljuk a már kiszámolt értékeket – memorizáljuk, s ha szükségünk van rájuk, használjuk fel őket!

A Fibonacci-számok

A rekurzív függvények matematikai elméletében éppúgy ismert, mint a biológiában a Fibonacci olasz matematikusról elnevezett számsorozat. Azt vizsgálta, hogy egy nyúlpár „alapította” nyúlnemzetség adott idő alatt mekkora létszámúra növekszik, figyelembe véve, hogy a leszármazottak is alaposan „besegítenek” a létszámnövelésbe.

Ha a szaporodás eléggé „szabályosan” történik, akkor az új generáció létszámát az előzőek ismeretében könnyen kiszámíthatjuk. Szerinte az új generáció növekedését az előző 2 generáció gyerekei teszik ki. E mögött az a feltételezés húzódik meg, hogy minden nyúlpár egyszerre éppen 2 utóddal járul a népességhez, és e „szokásukat” születésüket követő 2 egymásutáni időpontban „gyakorolják” (mert – mondjuk – mielőtt a harmadik szaporodásra sor kerülhetne, fazékba kerülnek).

Definíciója

$$Fib(n) = \begin{cases} 0 & ha\ \ n = 0 \\ 1 & ha\ \ n = 1 \\ Fib(n-1) + Fib(n-2) & ha\ \ n > 1 \end{cases}$$

Példa

a Fibonacci-szám sorozat első néhány tagja

```
0,1,1,2,3,5,8,13,21,34,55,...
```

```
Fib(N):
  Elágazás
    N=0 esetén Fib:=0
    N=1 esetén Fib:=1
    egyéb esetben Fib:=Fib(N-1)+Fib(N-2)
  Elágazás vége
Függvény vége.
```

A memorizálásos megoldásban F(i)≥0 jelenti, ha már kiszámoltuk az i-edik Fibonacci számot. A függvény hívása előtt az F vektor elemei értékét -1-re kell állítani!

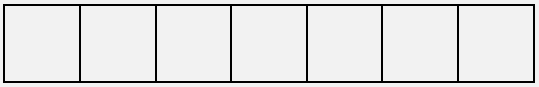
```
Fib(N):
  Ha F(N)<0 akkor Elágazás
                    N=0 esetén F(N):=0
                    N=1 esetén F(N):=1
                    egyéb esetben F(N):=Fib(N-1)+Fib(N-2)
                  Elágazás vége
  Elágazás vége
  Fib:=F(N)
Függvény vége.
```

Minta kódok.

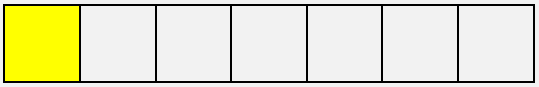
C++	cpp/fibonacci/feladat.cpp
C#	cs/fibonacci/feladat.cs
Java	java/fibonacci/feladat.java
Pascal	pas/fibonacci/feladat.pas
Python	py/fibonacci/feladat.py

Járdakövezés

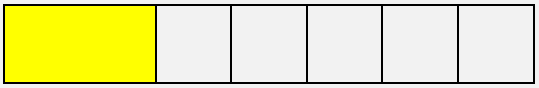
Számítsuk ki, hogy hányféleképpen lehet egy n egység méretű járdát kikövezni 1x1, 1x2 és 1x3 méretű lapokkal!



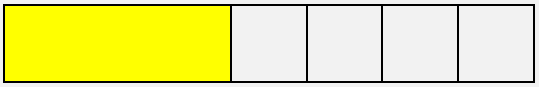
Az első helyre tehetünk 1x1-es lapot:



Az első helyre tehetünk 1x2-es lapot:



Az első helyre tehetünk 1x3-as lapot:



Az első esetben n-1, a másodikban n-2-t, a harmadikban pedig n-3 cellát kell még lefednünk. Azaz az n cella lefedéseinek Lefed(n) száma Lefed(n-1)+Lefed(n-2)+Lefed(n-3).

```
Lefed(N):
  Elágazás
    N=0 esetén Lefed:=0
    N=1 esetén Lefed:=1
    N=2 esetén Lefed:=2
    egyéb esetben Lefed:=Lefed(N-1)+Lefed(N-2)+Lefed(N-3)
  Elágazás vége
Függvény vége.
```

Itt már három rekurzív hívás van korábbi N értékekre, még nagyobb lehet az átfedés a kiszámolásokban. L(i)=-1 jelenti, hogy L(i)-t még nem számoltuk ki, L(i)≥0 jelenti, hogy már kiszámoltuk.

```
Lefed(N):
  Ha L(N)<0 akkor
     Elágazás
       N=0 esetén L(N):=0
       N=1 esetén L(N):=1
       N=2 esetén L(N):=2
       egyéb esetben L(N):=Lefed(N-1)+Lefed(N-2)+Lefed(N-3)
     Elágazás vége
  Elágazás vége
  Lefed:=L(N)
Függvény vége.
```

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Haladó
Téma	Rekurzív kiszámítás
Feladat	Járda 1 – 1x1,1x2,1x3

Binomiális együtthatók

Egy véges halmaz, melynek N darabszámú elemeiből K elemszámú halmazokat (kombinatorika nevén osztályokat) akarunk mindenféle módon képezni (és minden elem csak egyszer fordul elő). Ezt úgy hívjuk, hogy n elem k-ad osztályú ismétlés nélküli kombinációja. Ezen kombinációk száma megegyezik a matematikából máshonnan is ismert binomiális együtthatókkal.

A binomiális együtthatókat a következő képlettel definiálhatjuk:

$$\binom{n}{k} = \frac{n!}{k!*(n-k)!}$$

Ennek kiszámítása hosszú programot igényelne, próbálkozzunk inkább egy másik kiszámítási módszerrel!

Ehhez nézzük meg e számok elrendezését, a Pascal háromszöget:

```
                                    1
                                 1     1
                              1     2     1
                           1     3     3     1
                        1     4     6     4     1
                     1     5    10    10     5     1
                  1     6    15    20    15     6     1
               1     7    21    35    35    21     7     1
            1     8    28    56    70    56    28     8     1
         1     9    36    84   126   126    84    36     9     1
      1     10    45   120   210   252   210   120    45    10     1
   1    11    55   165   330   462   462   330   165    55    11    1
1     12    66   220   495   792   924   792   495   220    66    12    1
```

Felfedezhetjük, hogy a fenti táblázatban minden szám a fölötte levő két szám összege, azaz N elemből K elem választása leírható az alábbi módon:

- az első elemet választjuk, majd még N-1 elemből választunk K-1 elemet. vagy
- az első elemet nem választjuk és a maradék N-1 elemből választunk K elemet.

$$B(n,k) = \begin{cases} 1 & ha\ k = 0 \\ B(n-1,k) + B(n-1,k-1) & ha\ 0 < k < n \\ 1 & ha\ k = n \end{cases}$$

```
Bin(n,k):
  Ha k=0 vagy k=n akkor Bin:=1
  különben Bin:=Bin(n-1,k)+Bin(n-1,k-1)
Függvény vége.
```

Adott n és k értékre az alábbi téglalapbeli elemeket számoljuk ki, de mindegyiket annyiszor, ahányféleképpen az (n.k) elemből felfelé haladva eljuthatunk hozzájuk.

```
                                  1
                               1     1
                            1     2     1
                         1     3     3     1
                      1     4     6     4     1
                   1     5    10    10     5     1
                1     6    15    20    15     6     1
             1     7    21    35    35    21     7     1
          1     8    28    56    70    56    28     8     1
       1     9    36    84   126   126    84    36     9     1
    1     10    45   120   210   252   210   120    45    10     1
 1     11    55   165   330   462   462   330   165    55    11     1
1     12    66   220   495   792   924   792   495   220    66    12     1
```

Itt is memorizálunk, B(n,k)=-1 jelentse azt, hogy (n,k)-ra még nem számoltuk ki a Bin függvény értékét, B(n,k)≥0 pedig azt, hogy már kiszámoltuk:

```
Bin(n,k):
  Ha B(n,k)<0 akkor Ha k=0 vagy k=n akkor B(n,k):=1
                    különben B(n,k):=Bin(n-1,k)+Bin(n-1,k-1)
  Elágazás vége
  Bin:=B(n,k)
Függvény vége.
```

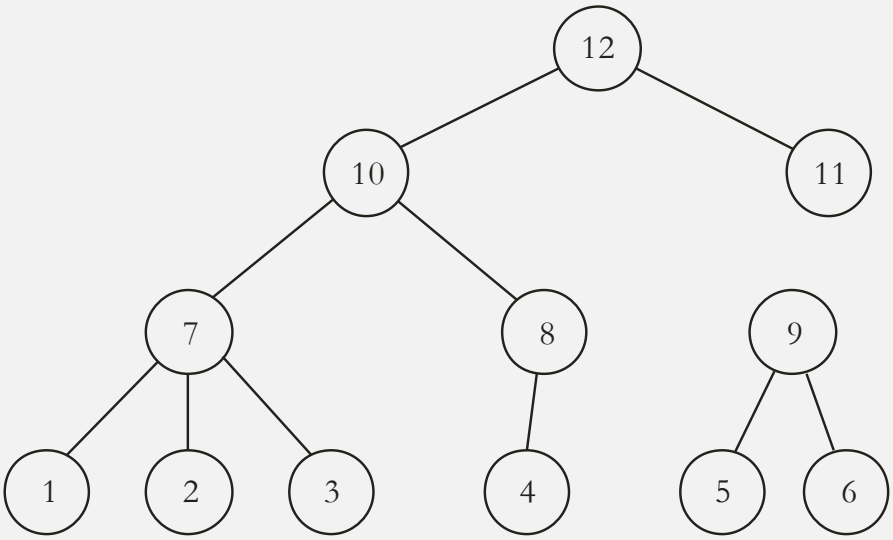
Minta kódok.

C++	cpp/binomialis/feladat.cpp
C#	cs/binomialis/feladat.cs
Java	java/binomialis/feladat.java
Pascal	pas/binomialis/feladat.pas
Python	py/binomialis/feladat.py

Ősök

N ember mindegyikéről tudjuk, hogy ki az anyja (a két szülőből így csak az egyiket kell tárolni), az embereket sorszámukkal azonosítjuk. Mindenkinek egyértelműen 1 anyja lehet, emiatt az adatokat egyetlen vektorban tárolhatjuk: Anya(i) jelenti az i. ember anyjának sorszámát. Akinek nem ismerjük az anyját, annál ez a sorszám legyen 0!

A kapcsolatokból egy ilyen hálózat épülhet fel:



Adjuk meg egy adott ember (A) legközelebbi ősét!

Az ős saját maga, ha az anyját nem ismerjük. Ha az anyját ismerjük, akkor a legrégebbi őse megegyezik az anyja legrégebbi ősével – megtaláltuk a rekurziót:

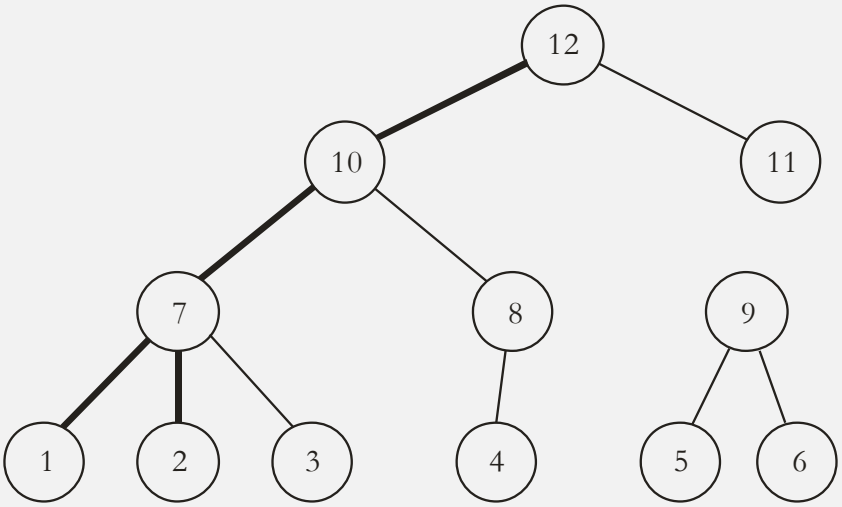
```
Ős(A):
  Ha Anya(A)=0 akkor Ős:=A
             különben Ős:=Ős(Anya(A))
Függvény vége.
```

Ez a rekurzív függvény csak egyszer hívja magát, azaz többször ugyanazzal a paraméterrel biztos nem hívja meg magát, nincs értelme memorizálással foglalkozni.

Módosítsuk a feladatot: adjuk meg két adott ember (A és B) legrégebbi közös ősét, ha van!

```
KözösŐs(A,B):
  x:=Őse(A)
  y:=Őse(B)
  Ha x=y akkor KözösŐs:=x
       különben KözösŐs:=0
Függvény vége.
```

Itt már felmerülhet a többszöri kiszámítás. Ha az előző példa szerint 1 és 2 közös ősét keressük, akkor az első Őse függvényhívás megadja a 12 sorszámú embert, a második függvényhívás pedig ebben a hálóban újra felmegy a 12-ig. A hálón azonban az látható, hogy ha másodszor a 7-ig elértünk, akkor már tudjuk, hogy a legrégebbi 12 a közös ős.



Tehát memorizáljuk A őseit!

```
Ős(A):
  Volt(A):=igaz
  Ha Anya(A)=0 akkor Ős:=A
             különben Ős:=Ős(Anya(A))
Függvény vége.

VanKözös(B):
  Ha Volt(B) akkor VanKözös:=igaz
  különben Ha Anya(B)=0 akkor VanKözös:=hamis
             különben VanKözös:=VanKözös(Anya(B))
Függvény vége.

KözösŐs(A,B):
  x:=Ős(A)
  Ha VanKözös(B) akkor KözösŐs:=x
                 különben KözösŐs:=0
Függvény vége.
```

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Haladó
Téma	Rekurzív adatszerkezetek
Feladat	63. Közös ősök

Ősök száma

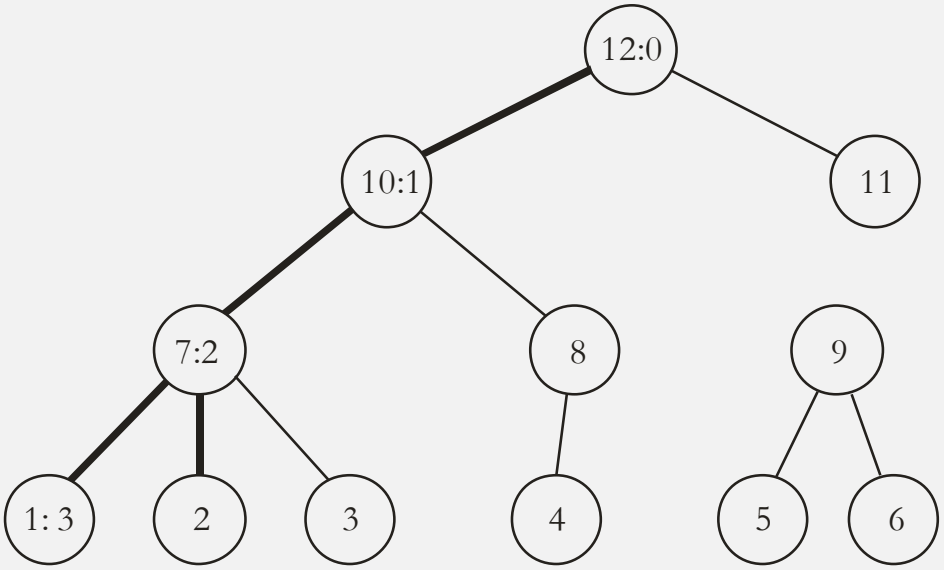
Számoljuk ki – az előző példát folytatva – egy adott ember (A) ősei számát!

Ha nem ismerjük az anyját, akkor ez a szám 0, ha ismerjük, akkor neki az anyja ősei számánál eggyel több őse van:

```
Ősszám(A):
  Ha Anya(A)=0 akkor Ősszám:=0
             különben Ősszám:=Ősszám(Anya(A))+1
Függvény vége.
```

Itt is egyetlen rekurzív hívás van, azaz memorizálásra nincs szükség.

Módosítsuk a feladatot, számoljuk ki két ember (A,B) közös ősei számát! A közös ősök halmaza az A és a B ősei halmazának közös része (metszete) lesz. Az alapmegoldásunk ennek meghatározására nem alkalmas – két ember ősei számából ez a szám nem határozható meg. Itt tehát mindenképpen – nem csak hatékonysági szempontból – szükségünk van memorizálásra.



A megoldásban az ősök számának meghatározása közben számoljuk ki az ősök ősei számát is!

```
Ősszám(A):
  Ha Anya(A)=0 akkor Db(A):=0
             különben Db(A):=Ősszám(Anya(A))+1
  Ősszám:=Db(A)
Függvény vége.
```

Ebben az esetben a B-vel közös ősök számát pontosan akkor tudjuk meg, amikor B-ből elérjük az első közös őst!

```
KözösŐsszám(B):
  Ha Db(B)≥0 akkor KözösŐsszám:=Db(B)+1
  különben Ha Anya(B)=0 akkor KözösŐsszám:=0
  különben KözösŐsszám:= KözösŐsszám(Anya(B))
Függvény vége.

KözösŐsökSzáma(A,B):
  x:=Ősszám(A)
  KözösŐsökSzáma:=KözösŐsszám(B)
Függvény vége.
```

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Haladó
Téma	Rekurzív adatszerkezetek
Feladat	Közös ősök száma

Ősök távolsága

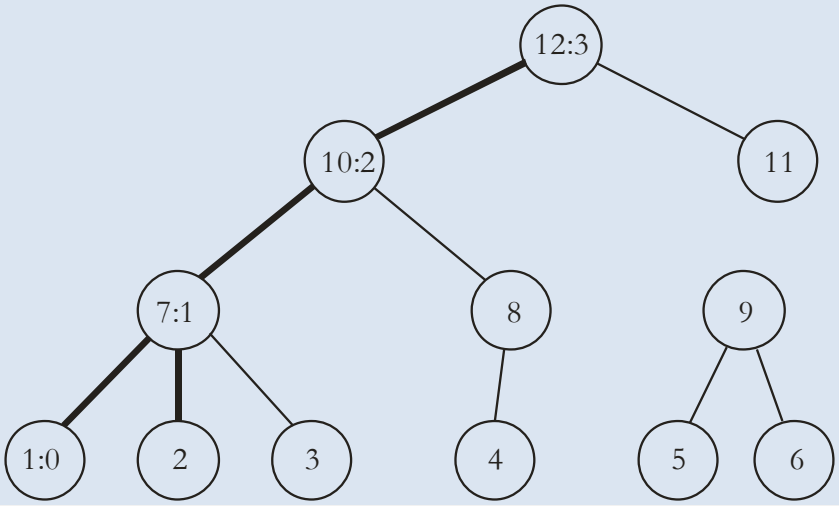
A következő feladatban azt szeretnénk tudni, hogy egy adott ember milyen „távolságra” van a legrégebbi ősétől. Nem meglepően ez ugyanaz a feladat, mint az ősök száma:

```
Távolság(A):
  Ha Anya(A)=0 akkor Távolság:=0
             különben Távolság:=Távolság(Anya(A))+1
Függvény vége.
```

Ha azonban arra lennénk kíváncsiak, hogy két ember legközelebbi közös őse milyen távol van tőlük (azaz a két távolság összege érdekel), akkor ez az információ (és ezzel együtt ez a megoldás) nem elégséges.

Memorizálnunk kell, de nem az ősök számát, hanem az adott embertől vett „távolságot”:

```
Távolság(A,i):
  TAV(A):=i
  Ha Anya(A)=0 akkor Távolság:=0
             különben Távolság:=Távolság(Anya(A),i+1)+1
Függvény vége.
```



Ekkor a közös ős távolság két távolság összegeként számolandó:

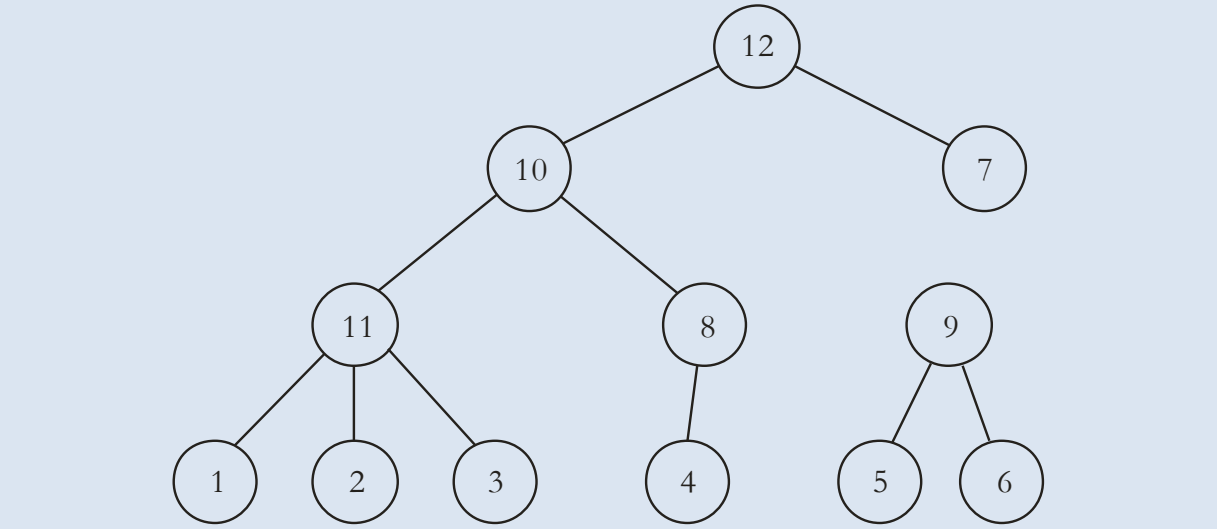
```
KözösTávolság(B,i):
  Ha TAV(B)≥0 akkor KözösTávolság:=TAV(B)+i
  különben Ha Anya(B)=0 akkor KözösTávolság:=-1
  különben KözösTávolság:=KözösTávolság(Anya(B),i+1)
Függvény vége.
```

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	???
Téma	???
Feladat	

Legtöbb ember őse

Ez az a feladat, ami csak memorizálással oldható meg – azt kell tárolni mindenkiről, hogy hány embernek őse. Praktikus lenne az emberek olyan sorszámozását megadni, ahol tudjuk, hogy kik azok, akik a hálóban alul vannak, mert belőlük érdemes elindulni felfelé.



Itt az 1..7, általánosan fogalmazva az 1..K sorszámú emberekből kell kiindulni.

```
Utódszám(A,x):
  Ha Anya(A)=0 akkor Db(A):=Db(A)+x
  különben Db(A):=Db(A)+x; Utódszám(Anya(A),x+1)
Eljárás vége.

LegtöbbUtód(max):
  Ciklus i=1-től K-ig
    Utódszám(i,0)
  Ciklus vége
  max:=1
  Ciklus i=2-től N-ig
    Ha Db(i)>Db(max) akkor max:=i
  Ciklus vége
Eljárás vége.
```

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	???
Téma	???
Feladat	