Az előző részben a 2014/15. évi Informatika OKTV Döntő 3. feladatára (Családfa, elérhető pl. http://nemes.inf.elte.hu/nemes_a adtunk megoldást. Az algoritmus minden gyermektelen személytől kiindulva bejárta a szülő–gyermek kapcsolatok által létrehozott gráfot, és megtalálta a közös ősöket. Mivel a feladatban akár 10 000 személy is szerepelhetett, ezért megpróbáltuk kiszámítani, hogy milyen nagyságrendű az algoritmus lépésszáma, nehogy nagy bemenetekre túllépjük az időkorlátot. Arra jutottunk, hogy a személyek N számának négyzetével arányos lépésszámú algoritmust készítettünk. A cikk végén kértük, hogy az olvasók maguk is töprengjenek jobb megoldáson, és írják meg a KöMaL honlapján működő Fórumban az Informatika OKTV témához.

A kisebb lépésszámú megoldás megtalálásához a futásidő becslése hasznos, mert közben alaposan elemezzük az algoritmusunkat, így láthatjuk, hogy melyek azok a részek, ahol érdemes javítani, hatékonyabb megoldást keresni. Szembetűnő, hogy a megoldásunk többször is bejárja azokat a részgráfokat, amelyek elérhetők egy-egy gyermektelen csúcstól. Próbáljuk meg elkerülni a gráf többszöri bejárását! Ez csak úgy lehetséges, ha minden érintett szülőnél nyilvántartjuk, hogy mely gyermektelenektől juthatunk el hozzá. A feladathoz mellékelt példában (ld. előző rész vagy a feladat linkje) induljunk ki a 4-es csúcstól, de csak a szülőkig haladjunk előre, és jegyezzük föl a 3-as és 8-as csúcshoz, hogy oda a 4-estől el lehet jutni. Ezután az 5-ös személy szüleivel folytassuk, és jegyezzük föl a 8-as és 9-es csúcshoz, hogy oda az 5-östől el lehet jutni. Amennyiben az összes gyermektelentől elértünk a szüleihez, és tároltuk a gyermekeik sorszámát, akkor ezután a szülőktől kell folytatnunk a bejárást. Most már azt kell kideríteni, hogy tőlük mely csúcsokba lehet eljutni. Tőlük folytathatjuk az előbbi módszerrel, de ügyeljünk arra, hogy egy csúcsból csak akkor térjünk át a szülő csúcsokhoz, ha a csúcshoz vezető összes gyermektől már odaértünk, és megjegyeztük, hogy mely gyermektelenek leszármazottai.

Az első megoldáshoz hasonló, de a eddigiektől eltérő vezérlésű bejárás kezd körvonalazódni. Minden lépésben egy gyermektől lépünk tovább a két szülőhöz, de csak olyan gyermektől, amelynek korábban minden gyermekéhez már eljutottunk. A gyermeknél lévő információt – vagyis az ő gyermektelen gyermekeinek sorszámát – átadjuk a két szülőnek. Így fokozatosan eljutunk az ősökhöz, és a bejárás végére kiderül, hogy mely ősöknél szerepel az összes gyermektelen sorszáma. A gráf bejárása tehát úgy történik, hogy minden egyes lépésnél egy olyan csúcsot választunk, amelyhez az összes oda vezető élt már bejártuk. Ilyen csúcs biztosan van az elején, tehát a sorba először a gyermektelenek kerülnek, majd az ő szüleik közül azok, amelyekhez már minden gyermekétől eljutottunk, és így tovább.

Nézzük, hogy milyen módon tároljuk a bejárás alatt az adatokat. Ha tudjuk, hogy g számú gyermektelen van, akkor $N-g$ számú szülőhöz kell fölvenni g számú logikai értéket, amely megadja, hogy az adott szülőt mely gyermektelenektől lehet elérni. A legegyszerűbb esetben $g \cdot (N-g)$ egybájtos szám tárolása szükséges. Ismét a számtani-mértani közepek közötti összefüggést alkalmazva $N^2/4$ logikai értékünk lesz, vagyis a legnagyobb elemszám esetén is 25 MiB alatti a helyfoglalás. Ez jelentős memóriaterület a korábban fölvett 10 KiB nagyságrendű adatokhoz képest.

Természetesen kevesebb hely is elegendő lenne, ha pl. minden csúcsnál egy listába fűznénk a hozzá vezető gyermektelenek sorszámát. De így minden él bejárásakor össze kellene vetni az él által összekapcsolt szülő és a gyermek gyermektelen listáit, ami a tömbös változatnál nagyobb lépésszámú algoritmust eredményezne. Vagy megtehetnénk, hogy bittömböket használunk, így 8-adára csökkenhető a tárolás. Megtehetnénk még azt is, hogy dinamikusan kezeljük a tömböket, vagyis mindig csak annyi tömbnek foglalunk helyet, amelyben tényleg van adat, hiszen a még el nem ért csúcsokban a táblázat csupa hamis értéket tartalmaz, és a már bejárt gyermekek adataira nincs is szükség. De egyelőre úgy számoltunk, hogy a memóriakorlát alatt vagyunk, ezért alkalmazzunk egyszerűbb adatszerkezetet. A legegyszerűbb persze egy $N \times N$-es tömb volna, de azzal már túllépnénk a memóriakorlátot.

Legyen tehát egy $N^2/4$ méretű `gy` tömbünk, amelyben logikai értékek vannak. Az első g darab szám megadja, hogy a bejárás során érintett első szülőhöz mely gyermektelenektől lehet eljutni. A második g darab szám megmutatja, hogy a második szülő mely gyermektelenektől érhető el stb. A `gy` tömb mellé szükségünk van még két másik táblázatra, amelyek megadják, hogy az adott sorszámú csúcs hányadik gyermektelen, illetve hányadik szülő. Ezekkel a számokkal már megcímezhetjük a `gy` tömb megfelelő elemét. Példaként legyen a feladathoz mellékelt gráf, amelyben négy gyermektelen van. Vegyünk föl egy N méretű `ind` tömböt, amely egyben tartalmazza a két táblázatot, hiszen egy csúcs vagy szülő, vagy gyermektelen. A tömb 4, 5, 6 és 7 sorszámú eleme rendre 1, 2, 3 és 4, mivel ez a négy csúcs a négy gyermektelent jelöli. A szülők az 1, 2, 3, 8, ..., 20 sorszámúak, az ő értékük a `gy` tömbben rendre 1, 2, 3, 4, ..., 16. Például a 8-as sorszámú (sorrendben 4. szülő) gyermeke az 5-ös (sorrendben 2.) gyermektelen, vagyis a táblázatában `gy[(ind[8]-1)*g+ind[5]]`, azaz `gy[(4-1)*4+2]`, tehát `gy[14]` értéke igaz. Általában tehát az `szk`-adik szülő `gyk`-adik gyermektelentől való elérhetőségét a `gy[(ind[szk]-1)*g+ind[gyk]]` érték adja meg.

Nézzük akkor a feladatot – remélhetőleg hatékonyabban – megoldó *`Családfa2`* algoritmust. Az adatok beolvasása és az `él` tömb létrehozása után több előkészítésre is szükség van. A bejárás minden lépésben egy olyan csúcsot választ, amely gyermektelen, vagy olyat, amely szülő, de már minden gyermekétől eljutottunk hozzá. Ez utóbbi megállapításához szükségünk van minden csúcsban arra az információra, hogy hány olyan él vezet hozzá, amelyet még nem jártunk be. Ezt a számot tárolhatjuk egy tömbben, amelynek megfelelő értékeit a bejárás közben mindig csökkentjük. Így ebben a táblázatban a 0 értékű, még fel nem dolgozott csúcsokon mindig folytatódhat a bejárás. A `jártunk` tömb helyett egy egészeket tároló `gysz` táblázatunk lesz, amely megmutatja, hogy hány gyerektől kell még elérnünk az adott csúcshoz. A gyermektelenek esetében az érték már kezdetben is 0. Az így létrehozott tömbök adatai alapján kitöltjük az `ind` táblázatot is, és feltöltjük a `gy` tömböt hamis értékekkel, hiszen még nem értünk el egyetlen csúcsot sem.

A bejárás során csak a szülőkkel rendelkező csúcsokból kell az éleken végighaladni, ezért tároljuk a szülő nélküli tulajdonságot is minden személynél egy `szn` logikai tömbben. Ez az érték ugyan azonnal látszik az `él` táblázatból, de

mégis érdemes külön nevet adni neki. Ők a lehetséges közös ősök, azok a személyek, akiket biztosan nem kell a sorba elhelyezni. Az adatok beolvasása után, de még a bejárás előtt gyűjtsük össze a bejáráshoz szükséges információkat és adjuk meg a fenti adatok kezdőértékeit:

```
   Családfa2_előkészítés
    gysz[] := 0
    szn[] := igaz
    gytn[] := hamis
    Ciklus személy := 1-től N-ig
        Ha él[személy][1]>0 akkor
            gysz[él[személy][1]] := gysz[él[személy][1]]+1
            gysz[él[személy][2]] := gysz[él[személy][2]]+1
            szn[személy] := hamis
        Elágazás vége
    Ciklus vége
    g = 0
    sz = 0
    Ciklus j := 1-től N-ig
        Ha gysz[j]=0 akkor g := g+1; ind[j] := g; gytn[j] := true
                     különben sz := sz+1 ; ind[j] := sz
        Elágazás vége
    Ciklus vége
    gy[] := hamis
Családfa2_előkészítés vége
```

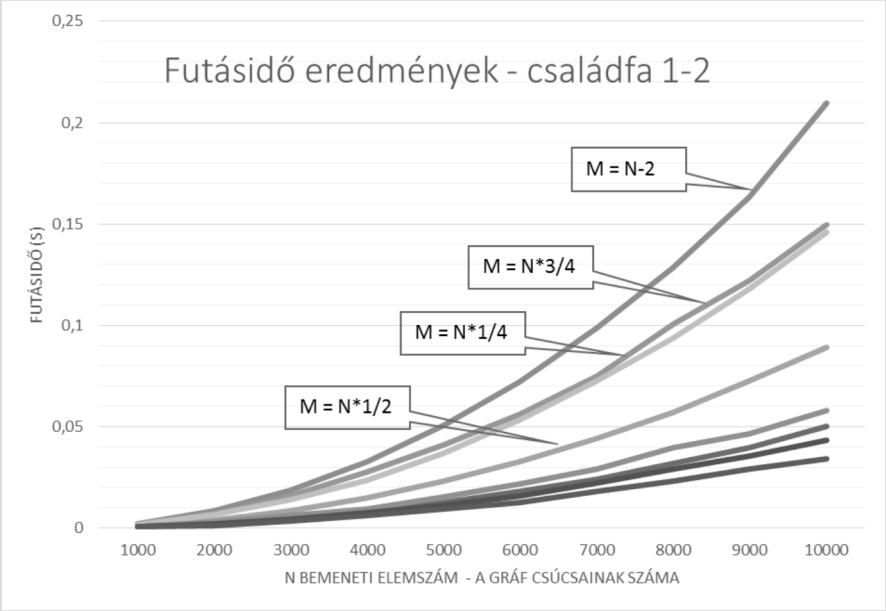
Az előkészítést végző algoritmus lépésszáma a három ciklusban és az N méretű tömbökben N-nel arányos, míg a `gy` tömb feltöltése ugyan N^2-tel, de ez a lépés egy egyszerű memóriafeltöltés azonos értékkel, amit a legtöbb rendszerben hatékonyan megvalósítanak.

Nézzük a bejárást, amely úgy kezdődik, hogy betesszük a sorba a gyermektelen, de ugyanakkor szülőként előforduló személyeket. Minden lépésben kiveszünk egy személyt a sorból, a `gysz` értékét a két szülőknél csökkentjük, és a szülők `gy` táblázatát a személyhez való elérhetőségekkel bővítjük. A sorba helyezünk minden olyan szülőt, aki nem ős, és az eddigiek során összegyűjtött minden információt, azaz minden hozzá vezető élt már bejártunk. A ciklus most is addig folytatódik, amíg van a sorban személy.

```
   Családfa2
    él_tömb_elkészítése
    családfa2_előkészítés
    Sor_legyen_üres
    Ciklus csúcs := 1-től N-ig
        Ha gysz[csúcs]=0 és nem szn[csúcs] akkor Sorba(csúcs)
    Ciklus vége
    Ciklus amíg Sor nem üres
        Sorból(személy)
        szülő1 := él[személy][1]
        szülő2 := él[személy][2]
        gysz[szülő1] := gysz[szülő1]-1
        gysz[szülő2] := gysz[szülő2]-1
        Ha gytn[személy] akkor
            gy[(ind[szülő1]-1)*g+ind[személy]] := igaz
            gy[(ind[szülő2]-1)*g+ind[személy]] := igaz
        különben
            Ciklus gyk := 1-től g-ig
                Ha gy[(ind[személy]-1)*g+gyk] akkor
                    gy[(ind[szülő1]-1)*g+gyk] := igaz
                    gy[(ind[szülő2]-1)*g+gyk] := igaz
                Elágazás vége
        Ciklus vége
        Ha gysz[szülő1]=0 és nem szn[szülő1] akkor Sorba(szülő1)
        Ha gysz[szülő2]=0 és nem szn[szülő2] akkor Sorba(szülő2)
    Ciklus vége
Családfa2 vége
```

A bejárás után nincs más dolgunk, mint végignézni a lehetséges ősök `gy` táblázatát, és amelyiknél minden bejegyzés igaz, azokat megszámoljuk, és sorrendben a kimenetre írjuk.

Gondoljuk végig, hogy a második megoldás valóban hatékonyabb lett-e az elsőnél. Az előkészítésről beszéltünk, most vizsgáljuk meg a bejárást. Bár összetettebb az egy csúccsal végzett feladat, mégis biztosan egyszer kerül be minden csúcs a sorba, tehát egyetlen élen sem fogunk kétszer áthaladni. Mivel a sorba $N - g$ számú csúcs kerül be, így most is a $g \cdot (N - g)$ szorzat mutatja a műveletek számát, amit szintén csak $N^2/4$-gyel tudunk felülről becsülni. Ennek ellenére azt várjuk, hogy a második megoldás az elsőnél hatékonyabb lesz. A futásidő minden olyan esetben nyilván kisebb, amikor sok a testvér. A futásidőt persze csak úgy kaphatjuk meg, hogy összehasonlítjuk a két algoritmus alapján készült programok futásidejét azonos, nagy számú bemenetekre. Az eredményt a *diagram* szemlélteti.



A négy nagyobb futásidőt mutató vonal a *Családfa1*, míg a kisebb futásidőt mutató négy vonal a *Családfa2* algoritmusok alapján készült programok futásidejét mutatja. Az eredményekből egyrészt leolvasható, hogy M értéke is jelentősen befolyásolja a programok futásidejét, ahogy ez természetesen várható is volt. Ugyanakkor jól látszik, hogy az algoritmusok lépésszáma a bemenet N elemszámának négyzetes függvénye. Az időmérés igazolta azt a várakozásunkat, hogy a második algoritmus gyorsabb, mint az első.

A feladat időkorlátjának megadott 0,5 másodperces futásidőt persze az OKTV döntőjében a programokat értékelő rendszer méri. A http://mester.inf.elte.hu oldalon elérhető az értékelést végző számítógép. Regisztráció, vagy Google azonosító segítségével használhatjuk az online programozási feladatbankot, amelyre ha feltöltjük a C/C++/Pascal vagy Java forrásprogramunkat, akkor lefordítja és futtatja azokat különböző tesztesetekre. Az értékelő rendszerben a az előző és a mostani részben közölt két algoritmus C++ változata leghosszabb futásideje 0,039 és 0,031 másodperc volt, vagyis az első programunkkal is maximális pontszámot szereztünk volna. Tapasztalatként szolgál, hogy egy legföljebb 10 000 elemszámú bemenetre kitűzött versenyfeladatra nem szükséges N^2 lépésszámú algoritmusnál hatékonyabbat készíteni.

A KöMaL Fórumon is született több megoldás, amelyeket érdemes lefuttatni és megnézni. A programok közreadására és futtatására használt http://ideone.com weboldal szintén tartalmaz egy online futtató rendszert, bár értékelést nem végez, és a bemeneteket is nekünk kell megadnunk. A rendszerről más esett szó a KöMaL-ban, a 2013. decemberi **I. 336.** feladatban egy róla szóló útmutató elkészítése volt a cél.

Feladatok és kérdések:

1. Készítsünk programot, amely a Családfa feladat egy helyes bemenetét adja, tehát a feladat leírásának megfelelő szülő–gyermek kapcsolatrendszert tartalmaz. A program bemenete N és M értéke legyen.

2. Készítsük el a második algoritmus bejárásának rekurzív változatát.

3. A Fórumban a [8]-as számú bejegyzéshez tartozó program szép és rövid (szerzője a Fórumban használt nevén Róbert Gida). Próbáljuk megérteni a programot, és találjuk ki, hogy mit végez benne a `fun()` függvény!