ÚJ SZÉCHENYI TERV

Ég és Föld vonzásában – a természet titkai

Informatikai tehetséggondozás:

Oszd meg és uralkodj stratégia

TÁMOP-4.2.3.-12/1/KONV

MAGYARORSZÁG MEGÚJUL

A projektek az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósulnak meg.

Egyik legfontosabb, sokféleképpen alkalmazható elvünk az ókori latin kultúrából ránk maradt *oszd meg és uralkodj* elve alapján fogalmazható meg: *oszd részekre*, majd a részek független megoldásával az egész feladatot könnyebben oldhatod meg. Így programod könnyen kézben tarthatod, vagyis *uralkodhatsz* felette.

Ezt a **stratégiai elv**et tartjuk szem előtt akkor, amikor gondolkodásmódunkat kívánjuk helyes mederbe terelni. *Lépésenkénti finomítás*nak nevezik ezt az elvet a feladatmegoldás filozófiájában. A feladatot néhány (=nem túl sok!) részfeladatra bontjuk. Úgy is mondhatnánk: a feladatot megoldjuk a legfelső szinten. Ezt az eljárást fogjuk követni az egyes részfeladatok megoldásakor is (a *részek feletti uralkodás* érdekében), mindaddig, amíg a részfeladatok elemi utasításokkal megoldhatóvá válnak.

A stratégia lépései:

- felosztás (megadjuk a részfeladatokat, amikre a feladat lebontható)
- uralkodás (rekurzívan megoldjuk az egyes részfeladatokat)
- összevonás (az egyes részfeladatok megoldásából előállítjuk az eredeti feladat megoldását)

Az ilyen feladatok megoldását természetéből adódóan rekurzívan fogalmazzuk meg. A rekurzív eljárás – mint a rekurzió minden esetben – kétféle részből kell álljon:

- a triviális esetből (amikor nincs rekurzív hívás), megadva ennek megoldását,
- az általános esetből, amikor rekurzívan visszavezetjük kisebb problémák megoldására (sokszor kettőre, de néha többre is).

Ezek alapján a következőképpen fogunk gondolkodni:

- Mi az általános feladat alakja? Mik a paraméterei? Ebből kapjuk meg a rekurzív eljárásunk specifikációját.
- Milyen paraméterértékekre kapjuk a konkrét feladatot? Ezekre fogjuk meghívni kezdetben az eljárást!
- Mi a leállás (triviális eset) feltétele? Hogyan oldható meg ilyenkor a feladat?
- Hogyan vezethető vissza a feladat hasonló, de egyszerűbb részfeladatokra? Hány részfeladatra vezethető vissza?
- Melyek ilyenkor az általános feladat részfeladatainak a paraméterei? Ezekkel kell majd meghívni a rekurzív eljárást!
- Hogyan építhető fel a részfeladatok megoldásaiból az általános feladat megoldása?

Logaritmikus keresés:

Ebben az esetben adott egy rendezett sorozat, amelyben egy adott értékű elem sorszámát kell meghatározni, ha az benne van a sorozatban.

A rendezettséget (és az indexelhetőséget) kihasználhatjuk. Vizsgáljuk meg első lépésben a sorozat középső elemét! Ha ez a keresett elem, akkor készen vagyunk. Ha a keresett elem ennél kisebb, akkor csak az ezt megelőzőek között lehet, tehát a keresést a továbbiakban a sorozatnak arra a részére kell alkalmazni. Ha a keresett ennél nagyobb, akkor pedig ugyanezen elv alapján a sorozat ezt követő részére.

- leállási feltétel: az éppen vizsgált sorozatnak nincs eleme, ekkor nincs a keresett elem a sorozatban
- felbontás: a sorozat két részsorozatra bontása (középen)
- $\boxed{X_1, \ldots X_{k-1}}$, X_k $\boxed{X_{k+1}, \ldots, X_n}$
- uralkodás: az egyik részsorozatban tovább keresés, ha kell (rekurzívan)
- összevonás: nincs vele teendő, az eredmény kész

```
Keresés(X,E,U,Y,Van,K):
  Ha E>U akkor Van:=hamis
  különben K:=(E+U)/2
           Ha X(K)=Y akkor Van:=igaz
           különben ha X(K)<Y akkor Keresés(X,K+1,U,Y,Van,K)
           különben {ha X(K)>Y akkor} Keresés(X,E,K-1,Y,Van,K)
  Elágazás vége
Eljárás vége.
```

A lényeg tehát: válasszuk ki a középső elemet, s ennek vizsgálata alapján döntsük el, hogy az előtte vagy a mögötte levők között kell-e tovább keresni!

Az átlagos hasonlítás-számban szereplő logaritmikus összefüggés miatt hívják e keresést *logaritmikus keresés*nek, illetve a sorozat felezésére utalva *felezéses* vagy *bináris keresés*nek.

Rendezett mátrixban keresés

Ha nem vektorban, hanem rendezett mátrixban kell keresnünk, akkor a megoldás nagyon hasonló lesz:

1	3	4	7
8	11	13	16
16	16	20	25

Első lépésként keressük meg azt a sort, ahol az elem előfordulhat (logaritmikusan), majd az adott sorban keressük meg az elemet (szintén logaritmikusan)!

```
Sorkeresés(X,E,U,Y,Van,K,L):
  Ha E>U akkor Van:=hamis különben
     K:=(E+U)/2
     Ha X(K,1)≥Y és X(K,M)≤Y akkor Keresés(X,1,M,Y,Van,K,L)
     különben ha X(K,M)<Y akkor Sorkeresés(X,K+1,U,Y,Van,K,L)
     különben Sorkeresés(X,E,K-1,Y,Van,K,L)
  Elágazás vége
Eljárás vége.

Keresés(X,E,U,Y,Van,K,L):
  Ha E>U akkor Van:=hamis különben
         L:=(E+U)/2
         Ha X(K,L)=Y akkor Van:=igaz
         különben ha X(K,L)<Y akkor Keresés(X,L+1,U,Y,Van,K,L)
         különben Keresés(X,E,L-1,Y,Van,K,L)
  Elágazás vége
Eljárás vége.
```

Hiányzó érték keresése

Az 1,2,…,N+1 növekvő sorozatból egyetlen szám hiányzik. Melyik?

- felbontás: a sorozat két részsorozatra bontása (középen)
- $X_1, ..., X_k$ $X_{k+1}, ... , X_n$
- leállási feltétel: ha X_k nem k értékű, de X_{k-1} k-1 értékű, akkor a k érték hiányzik a sorozatból
- uralkodás: ha X_k=k, akkor a sorozat második feléből hiányzik egy érték, különben az első feléből (rekurzívan)
- összevonás: nincs teendő, valamelyik ágon eredményt kapunk.

```
Hiányzó(X,E,U,Y):
  K:=(E+U)/2
  Ha X(K)≠K akkor Ha X(K-1)=K-1 akkor Y:=K
                  különben Hiányzó(X,E,K-1,Y)
  különben Hiányzó(X,K+1,U,Y)
  Elágazás vége
Eljárás vége.
```

Párhuzamos maximum-minimum kiválasztás

Ha egyszerre kell egy sorozat maximumát és minimumát is meghatározni, akkor a maximumkiválasztás tételénél az oszd meg és uralkodj stratégia gyorsabb megoldást adhat. A tétel szerint a maximum és a minimum kiválasztása is N-1 lépésben történhet meg.

A megoldás ötlete: 2 elem közül 1 hasonlítással eldönthetjük, hogy melyik a maximum és melyik a minimum. Ha kettőnél több elemünk van, akkor osszuk két részre a sorozatot, mind-

két részben határozzuk meg a maximumot és a minimumot, majd ezekből adjuk meg a teljes sorozat maximumát és minimumát!

- leállási feltétel: az éppen vizsgált sorozatnak legfeljebb 2 eleme van: a maximum és a minimum 1 hasonlítással meghatározható
- felbontás: a sorozat két részsorozatra bontása (középen)
- $X_1, \ldots X_{k-1}, X_k \mid X_{k+1}, \ldots, X_n$
- uralkodás: mindkét részsorozatra meghatározzuk a maximumot és a minimumot (rekurzívan)
- összevonás: a két maximum közül a nagyobb lesz a sorozat maximuma, a két minimum közül pedig a kisebb lesz a sorozat minimuma.

Itt már csak 3*(n/2)-2 összehasonlításra van szükségünk 2*n-2 helyett.

```
Maxmin(X,E,U,Max,Min):
  Ha U-E=0 akkor Max:=E; Min:=E
  különben ha U-E=1 akkor
     Ha X(E)≤X(U) akkor Max:=U; Min:=E
     különben Max:=E; Min:=U
  különben K:=(E+U)/2
           Maxmin(X,E,K,Max1,Min1)
           Maxmin(X,K+1,U,Max2,Min2)
           Ha X(Min1)≤X(Min2) akkor Min:=Min1
                              különben Min:=Min2
           Ha X(Max1)≥X(Max2) akkor Max:=Max1
                              különben Max:=Max2
  Elágazás vége
Eljárás vége.
```

N darab szám legnagyobb közös osztója

A legnagyobb közös osztót meghatározó eljárás 2 számra működik, futási ideje a két szám összegével arányos (azaz annál gyorsabb, minél kisebbek a számok).

Vehetnénk először az első két szám legnagyobb közös osztóját, majd az így kapott szám és a harmadik legnagyobb közös osztóját, … és így tovább.

Elindulhatunk más elképzeléssel is: a sorozatot bontsuk két részre; mindkét résznek határozzuk meg a legnagyobb közös osztóját, majd ezek legnagyobb közös osztója lesz a megoldás. Ehhez a legnagyobb közös osztó alábbi tulajdonságát használjuk ki:

$\text{lnko}(X_1, \ldots X_k, X_{k+1}, \ldots, X_n) = \text{lnko}(\text{lnko}(X_1, \ldots X_k), \text{lnko}(X_{k+1}, \ldots, X_n))$

- leállási feltétel: az éppen vizsgált sorozatnak 1 eleme van: a legnagyobb közös osztó önmaga
- felbontás: a sorozat két részsorozatra bontása (középen)

- $\boxed{X_1, ..., X_k}$ $\boxed{X_{k+1}, ... , X_n}$
- uralkodás: mindkét részsorozatra meghatározzuk a legnagyobb közös osztót (rekurzívan)
- összevonás: a két legnagyobb közös osztónak vesszük a legnagyobb közös osztóját.

Gyorsítási lehetőségek

- ha az első rész legnagyobb közös osztója 1, akkor a második részt már ki sem kell számolni, az eredmény 1;
- ha a második rész legnagyobb közös osztója 1, akkor a két rész legnagyobb közös osztója is biztosan 1.

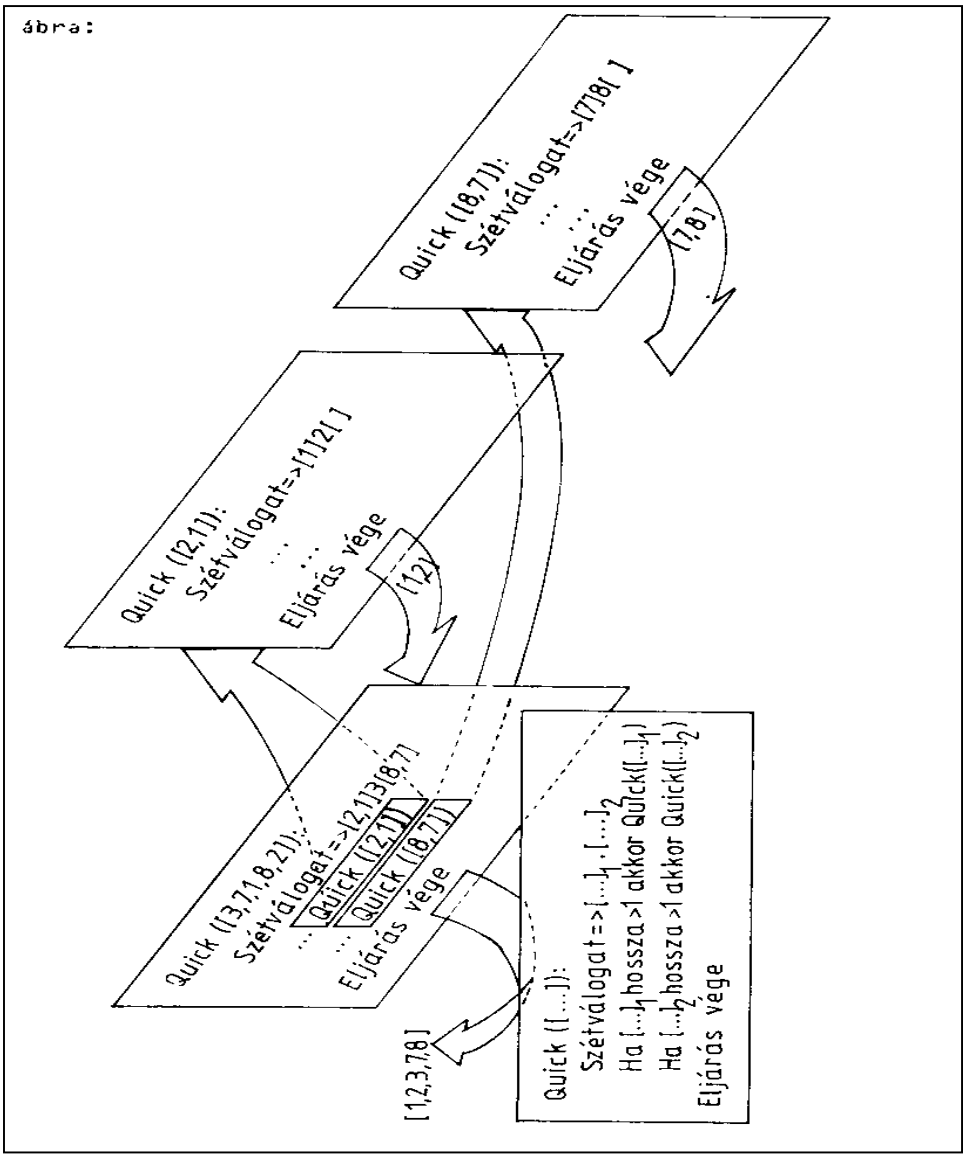
```
Legnagyobbközösosztó(X,E,U,L):
  Ha U-E=0 akkor L:=X(E)
  különben K:=(E+U)/2; L:=1
           Legnagyobbközösosztó(X,E,K,L1)
           Ha L1>1 akkor Legnagyobbközösosztó(X,K+1,U,L2)
                         Ha L2>1 akkor L:=lnko(L1,L2)
  Elágazások vége
Eljárás vége.
```

Gyorsrendezés (quicksort)

Lényege: Válogassuk szét úgy az X vektort, hogy az aktuális első elemnél kisebbek az elem elé kerüljenek, a nagyobbak pedig mögé. S mind az előtte, mind a mögötte levő részre végezzük el a fenti eljárást! Az 1, illetve 0 elemszámú intervallum már rendezett.

- leállási feltétel: a sorozat legfeljebb 1 elemű, ekkor definíció szerint rendezett;
- felbontás: $\boxed{X_1, ... , X_{k-1}}$ X_k $\boxed{X_{k+1}, ... , X_n}$ szétválogatás (ahol $\forall i,j$ $(1 \le i < k;\ k < j \le n)$: $X_i \le X_j$)
- uralkodás: mindkét részt ugyanazzal a módszerrel felbontjuk két részre, rekurzívan
- összevonás: automatikusan történik a helyben szétválogatás miatt

```
Quick(E,U):
  Szétválogatás(E,U,K)
  Ha E<K-1 akkor Quick(E,K-1)
  Ha k+1<U akkor Quick(K+1,U)
Eljárás vége.
```



A szétválogatást úgy végezzük el, hogy a tömb első elemét kivesszük a helyéről. Az utolsó elemtől visszafelé keresünk egy olyat, amely kisebb a kivett elemnél, s ezt előre hozzuk a kivett elem helyére. Ezután a hátul felszabadult helyre elölről keresünk egy a kivettnél nagyobb elemet, s ha találtunk azt hátratesszük. Mindezt addig végezzük, amíg a tömbben két irányban haladva össze nem találkozunk. Ekkor az elsőnek kivett elemet betesszük az üres helyre.

```
Szétválogatás(N,X,K):
  E:=1; U:=N; segéd:=X(E)
  Ciklus amíg E<U
    Ciklus amíg E<U és segéd≤X(U)
      U:=U-1                    [kivettnél kisebb elem keresése]
    Ciklus vége
    Ha E<U akkor X(E):=X(U); E:=E+1
                 Ciklus amíg E<U és segéd≥X(E)
                   E:=E+1  [kivettnél nagyobb elem keresése]
                 Ciklus vége
                 Ha E<U akkor X(U):=X(E); U:=U-1
    Elágazás vége
  Ciklus vége
  X(E):=segéd; K:=E
Eljárás vége.
```

Az i-edik legkisebb kiválasztása

A gyorsrendezés az alsorozat elemeit két részre osztotta: a kijelölt elemnél kisebb elemek és annál nagyobbak sorozatára. Eme sorozatok hossza alapján dönthetünk, hogy mely sorozatban található a keresett elem. Ezt a módszert követve átlagosan lineáris időben kereshetjük meg az i. elemet.

- felbontás: $\boxed{X_1, \ldots, X_{k-1}}$ X_k $\boxed{X_{k+1}, \ldots, X_n}$ szétválogatás (ahol $\forall i,j$ $(1 \le i \le k; k \le j \le n)$: $X_i \le X_j$)
- leállási feltétel: i=K esetén megtaláltuk a keresett elemet;
- uralkodás: i<K esetén az első, i>K esetén a második részben keresünk tovább, rekurzívan
- összevonás: automatikusan történik a helyben szétválogatás miatt

```
Kiválasztás(E,U,i,Y):
  Szétválogatás(E,U,K)
  Ha i=K akkor Y:=X(K)
  különben ha i<K akkor Kiválasztás(E,K-1,i,Y)
  különben Kiválasztás(K+1,U,i-K,Y)
Eljárás vége.
```

Nem létező elem keresése

Egy szekvenciális file-ban 1 és M (pl. M=1 000 000 000) közötti egész számok találhatók. 1 000 000 000 szám nem fér be a számítógép központi memóriájába. Adjunk meg egy olyan számot, ami nem szerepel ebben a szekvenciális állományban!

A legegyszerűbb megoldásban legfeljebb 1 000 000 000-szor végigolvassuk az állományt, s az első olyan számnál, amit nem találtunk az állományban, megállunk. Azt kell észrevenni,

hogy a felezendő intervallumot nem az állományban levő számok határozzák meg, hanem az az intervallum, ahol a keresett szám lehet. Így a megoldásban számoljuk le, hogy hány szám van 1 és 500 000 000 között, illetve 500 000 001 és 1 000 000 000 között, s amelyik intervallumban kevesebbet találtunk, abban biztosan van még fel nem használt szám, tehát arra az intervallumra alkalmazzuk ugyanezt az eljárást!

- felbontás: 1, ..., K K+1, ..., M számlálás (a potenciális elemértékekre)
- leállási feltétel: ha valamelyik részben a darabszám 0, akkor onnan választunk egy elemet;
- uralkodás: amelyik intervallumban kevesebb elem van, abban keresünk tovább, rekurzívan
- összevonás: automatikusan történik a helyben szétválogatás miatt

```
Nemlétező(E,U,Y):
  K:=(E+U)/2
  Számlálás(E,K,U,A,B)
  Ha A=0 akkor Y:=E
  különben ha B=0 akkor Y:=U
  különben ha A>B akkor Nemlétező(K+1,U,Y)
  különben Nemlétező(E,K-1,Y)
Eljárás vége.
```

Összefésüléses rendezés (mergesort):

Ha van két rendezett sorozatunk, akkor abból az összefésülés algoritmusa gyorsan elő tudja állítani a két sorozat elemeit tartalmazó rendezett sorozatot:

```
Összefésülés(N,X,M,Y,DB,Z):
  I:=1; J:=1; DB:=0
  X(N+1):=+∞; Y(M+1):=+∞ [az elemtípus maximális értéke]
  Ciklus amíg I<N+1 vagy J<M+1
    DB:=DB+1
    Ha X(I)<Y(J) akkor Z(DB):=X(I); I:=I+1
                 különben Z(DB):=Y(J); J:=J+1
  Ciklus vége
Eljárás vége.
```

Az egyelemű sorozatok mindig rendezettek, az N elemű sorozat pedig mindig N darab 1-elemű sorozatból áll. Ez megalapozhat egy rendező módszert: válasszuk a rendezendő sorozatot két részre, mindkét részt rendezzük, majd a két rendezett sorozatot fésüljük össze!

- leállási feltétel: ha a sorozat legfeljebb 1 elemű, az biztosan rendezett
- felbontás: a sorozat két részsorozatra bontása (középen): $X_1, \ldots, X_k$ $X_{k+1}, \ldots, X_n$
- uralkodás: a két részsorozat rendezése (rekurzívan)

- összevonás: a két rendezett részsorozat összefésülése

```
Rendez(X,E,U):
  Ha E<U akkor K:=(E+U)/2
               Rendez(X,E,K); Rendez(X,K+1,U)
               Összefésül(X,E,K,U)
Eljárás vége.
```

Az egyetlen probléma: az összefésülés megvalósítása helyben nagyon bonyolult, ezért az összefésülendőket először lemásoljuk.

```
Összefésülés(X,E,K,U):
  I:=1; J:=1; DB:=E-1; Y():=X(E..K); Z():=X(K+1..U)
  Y(K-E+2):=+∞; Z(U-K+1):=+∞ [az elemtípus maximális értéke]
  Ciklus amíg I<K-E+2 vagy J<U-K+1
    DB:=DB+1
    Ha Y(I)<Z(J) akkor  X(DB):=Y(I); I:=I+1
               különben X(DB):=Z(J); J:=J+1
  Ciklus vége
Eljárás vége.
```

Inverziók száma:

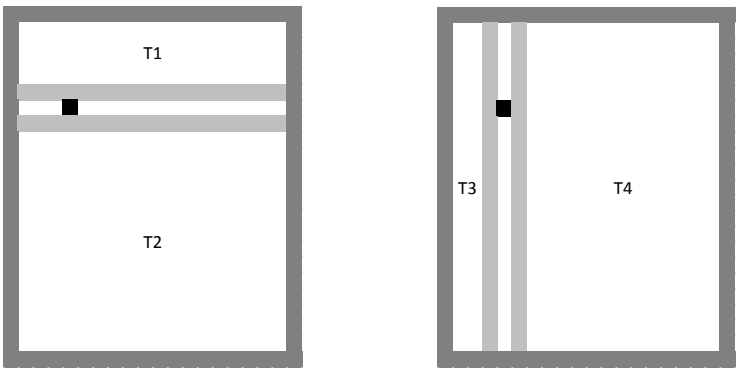
Az inverziók száma egy sorozatban azon (i,j) elempárok száma, ahol i<j és X(i)>X(j). A megoldás nagyon hasonló az összefésüléses rendezéshez. az inverziók száma = az inverziók száma az első félben + az inverziók száma a második félben + az inverziók száma a két fél között (a megoldás közben pedig rendezünk)

- leállási feltétel: ha a sorozat legfeljebb 1 elemű, az inverziók száma 0
- felbontás: a sorozat két részsorozatra bontása (középen): $X_1, \ldots, X_k$ $X_{k+1}, \ldots, X_n$
- uralkodás: a két részsorozat rendezése és inverzió számlálása (rekurzívan)
- összevonás: a két rendezett részsorozat összefésülése és inverzió számlálása, majd a három inverziószám összeadása

```
Rendez_számol(X,E,U,DB):
  Ha E<U akkor K:=(E+U)/2
               Rendez_számol(X,E,K,A)
               Rendez_számol(X,K+1,U,B)
               Összefésül_számol(X,E,K,U,C)
               DB:=A+B+C
  különben DB:=0
Eljárás vége.
```

Legnagyobb üres téglalap

A sík A…B (egész koordinátájú) pontok által határolt téglalapjában N (szintén egész koordinátájú) pont található. A feladat a legnagyobb résztéglalap keresése, amely egyetlen pontot sem tartalmaz! Egy megvizsgálandó téglalapot minden belső pont négy megvizsgálandó részre vág.



- leállási feltétel: ha a pont nincs az adott téglalapon belül, akkor az egész téglalap üres
- felbontás: a téglalap négy téglalapra bontása: az adott ponttól felfelé, lefelé, balra, illetve jobbra levő téglalapra (ha van)
- uralkodás: a négy téglalapra a legnagyobb üres téglalap számítása a maradék pontok alapján (rekurzívan)
- összevonás: a négy üres téglalap területéből maximumszámítás.

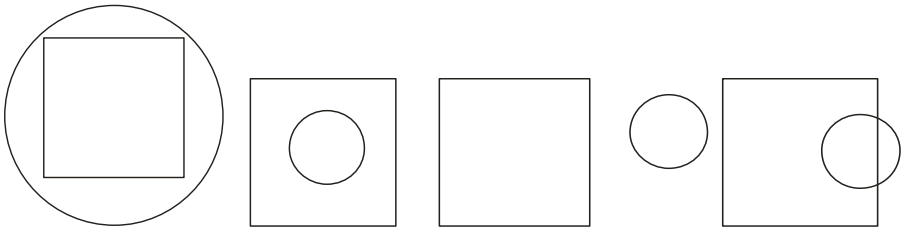
```
Téglalap(A,B,N,P,C,D,T):
  Ha belül(A,B,P(N)) akkor
    Ha A.x=P(N).x akkor T:=0
    különben B1.x:=P(N).x-1; B1.y:=B.y
             Téglalap(A,B1,N-1,P,C,D,T)
    Ha B.x=P(N).x akkor T2:=0
    különben A2.x:=P(N).x+1; A2.y:=A.y
             Téglalap(A2,B,N-1,P,C2,D2,T2)
    Ha T2>T akkor T:=T2; C:=C2; D:=D2
    Ha A.y=P(N).y akkor T3:=0
    különben B3.y:=P(N).y-1; B3.x:=B.x
             Téglalap(A,B3,N-1,P,C3,D3,T3)
    Ha T3>T akkor T:=T3; C:=C3; D:=D3
    Ha B.y=P(N).y akkor T4:=0
    különben A4.y:=P(N).y+1; A4.x:=A.x
             Téglalap(A4,B,N-1,P,C4,D4,T4)
    Ha T4>T akkor T:=T4; C:=C4; D:=D4
  különben C:=A; D:=B; T:=(B.x-A.x+1)*(B.y-A.y+1)
  Elágazás vége
Eljárás vége.
```

Négyzet és kör

A sík egy H oldalhosszúságú négyzete (bal alsó sarka a P pont) és egy Q középpontú R sugarú kör közös területének megadása egész pontossággal. (Minden adat legyen egész értékű!)

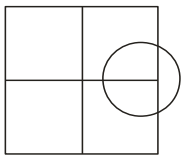
Egy négyzet és egy kör egymáshoz viszonyítva négy különböző helyzetben lehet:

- a négyzet része a körnek,
- a kör része a négyzetnek,
- a négyzet a körön kívül van,
- a négyzet átfed valamennyit a körből.



Az első esetben a közös rész a teljes négyzet, a másodikban pedig a teljes kör. A harmadik esetben nincs közös részük.

A negyedik esetben a négyzet metszi a kört, ekkor az előző feladathoz hasonlóan négy részre vágjuk, csak most egyformára. A négy négyzetre rekurzívan ugyanezt az eljárást folytatjuk, amíg a négyzetek területe 1-nél kisebbé nem válik.



- leállási feltétel: az első három eset, valamint ha a négyzet oldalhossza kisebb lesz 1-nél
- felbontás: a négyzet négy egyenlő négyzetre bontása
- uralkodás: a négy négyzetre a közös terület kiszámítása (rekurzívan)
- összevonás: a négy közös terület összeadása.

```
Közös(P,H,Q,R,T):
  Ha H<1 akkor T:=0
  különben E:=Esetválasztás(P,H,Q,R)
           Ha E=1 akkor T:=H*H
           különben ha E=2 akkor T:=R*R*π
           különben ha E=3 akkor T:=0
           különben Közös(P,H/2,Q,R,T1)
                    P.x:=P.x+H/2; Közös(P,H/2,Q,R,T2)
                    P.y:=P.y+H/2; Közös(P,H/2,Q,R,T3)
                    P.x:=P.x-H/2; Közös(P,H/2,Q,R,T4)
                    T:=T1+T2+T3+T4
Eljárás vége.
```

Gyorsítási lehetőség: ha a negyedik eset egyszer előfordult, utána a második eset már biztosan nem lehetséges, azaz ennek vizsgálatát a rekurzív eljáráson kívülre tehetjük.

Karatsuba szorzási algoritmusa

Ha két nagyon sok számjegyből álló kettes számrendszerbeli számot kell összeszorozni, akkor a következőképpen járhatunk el:

Legyen $X=X_1*2^N+X_2$, $Y=Y_1*2^N+Y_2$! Ekkor a szorzat: $X*Y=(X_1*2^N+X_2)*(Y_1*2^N+Y_2)= X_1*Y_1*2^{2*N}+X_1*Y_2*2^N+X_2*Y_1*2^N+X_2*Y_2= X_1*Y_1*2^{2*N}+(X_1*Y_2+X_2*Y_1)*2^N+X_2*Y_2$.

Azaz minden szorzás visszavezethető feleakkora számok 4 szorzására, valamint eltolásokra. Ügyes átrendezéssel azonban három szorzás is elég lehet:

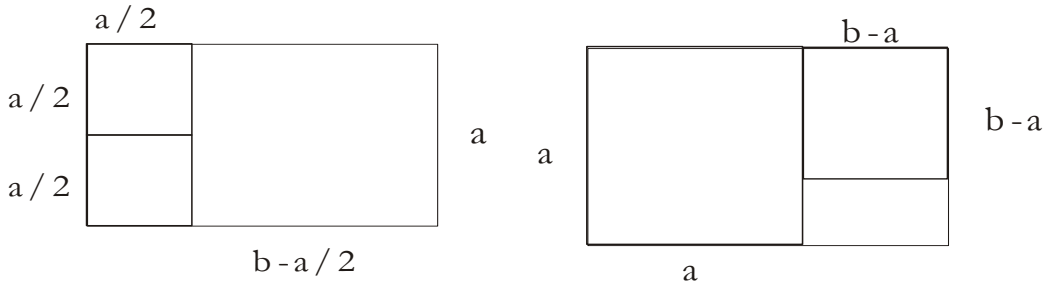
Ha kiszámoljuk $(X_1+X_2)*(Y_1+Y_2)$ értékét, akkor a következőt kapjuk:

$X_1*Y_1+X_1*Y_2+X_2*Y_1+X_2*Y_2$, ha ebből levonjuk X_1*Y_1 és X_2*Y_2 értékét – amit már kiszámoltunk – akkor éppen a fenti képletben szereplő zárójeles kifejezést kapjuk.

Téglalap felbontás

Adott egy a×b méretű téglalap (b−a<a<b).

Egy ilyen alakzat kétféleképpen bontható két négyzetre és egy téglalapra:



A következő lépésben a kapott téglalapot tovább bontjuk, és ezt addig folytatjuk, míg felbonthatatlan téglalapot nem nyerünk (nem elégítik ki a fenti összefüggést). Határozzuk meg a minimális lépésszámot, amellyel felbonthatatlan téglalaphoz jutunk!

- leállási feltétel: b−a<a<b nem teljesül
- felbontás: a kétféle felbontás két négyzetre és egy téglalapra
- uralkodás: a téglalapokra a minimális lépésszám kiszámolása (rekurzívan)
- összevonás: a két lépésszám minimumának meghatározása.

```
Felbontás(A,B,L):
  Ha B-A≥A vagy A≥B akkor L:=0
  különben Ha A<B-A/2 akkor Felbontás(A,B-A/2,L1)
                        különben Felbontás(B-A/2,A,L1)
           Ha B-A<2*A-B akkor Felbontás(B-A,2*A-B,L2)
                        különben Felbontás(2*A-B,B-A,L2)
           Ha L1<L2 akkor L:=L1 különben L:=L2
Eljárás vége.
```