



Belépő a tudás közösségébe

Informatika szakköri segédanyag



Oszd meg és uralkodj stratégia

**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



BEFÉKTETÉS A JÖVŐBE

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 elvet** tartjuk szem előtt akkor, amikor gondolkodás módunkat 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 kétfőre, 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)
- $\boxed{X_1, \dots, X_{k-1}}, X_k, \boxed{X_{k+1}, \dots, X_n}$
- uralkodás: az egyik részsorozatban tovább keresés, ha kell (rekurzívan)
- összevonás: nincs vele teendő, az eredmény kész

N elemű X tömbben Y-t keressük (bemenet):
van-e benne (Van), ha igen, akkor melyik (K) (kimenet)

Keresés(X, Y, Van, K) :
[N=X.elemszama]
Keresés(X, 1, N, Y, Van, K)

Eljárás vége.

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 döntünk el, hogy az előtte vagy a mögötte levők között kell-e tovább keresni!

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

Minta kódok.

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



Rendezett mátrixban keresés

Ha nem vektorban, hanem rendezett mátrixban kell keresnünk (pl.: a postás a postaládáknál), 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)!

N elemű X mátrixban Y-t keressük:
van-e benne (Van), ha van, akkor melyik sorban (K), melyik mezőben (L)

Keresés (X, Y, Van, K, L) :
Sorkeresés (X, 1, N, Y, Van, K, L)

Eljárás vége.

Sorkeresés (X, E, U, Y, Van, K, L) :

Ha $E > U$ **akkor** Van:=hamis **különben**

$K := (E + U) / 2$

Ha $X(K, 1) \leq Y$ és $Y \leq X(K, M)$ **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.

Minta kódok.

C++ cpp/rendmatker/feladat.cpp

C# cs/rendmatker/feladat.cs

Java java/rendmatker/feladat.java

Pascal pas/rendmatker/feladat.pas

Python py/rendmatker/feladat.py



Hiányzó érték keresése

Az $1, 2, \dots, N+1$ növekvő sorozatból egyetlen szám hiányzik, azaz a sorozat N elemű

- felbontás: a sorozat két részsorozatra bontása (középen)
- $\boxed{X_1, \dots, X_k} \boxed{X_{k+1}, \dots, X_n}$
- 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.

N elemű X tömbben melyik Y indexű elem a hiányzó

Hiányzó(X, Y) :

Hiányzó(X, 1, N, Y)

Eljárás vége.

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)

Elágazás vége

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, mindkét 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)
- $\boxed{X_1, \dots, X_{k-1}, X_k} \quad \boxed{X_{k+1}, \dots, X_n}$
- 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 $3 \cdot (n/2) - 2$ összehasonlításra van szükségünk $2 \cdot n - 2$ helyett.

N elemű X tömbből visszkapjuk a Max és Min értékeket

Maxmin(X, Max, Min) :

Maxmin(X, 1, N, Max, Min)

Eljárás vége.

```

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 (LNKO) 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:

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

- leállási feltétel: az éppen vizsgált sorozatnak 1 eleme van: a legnagyobb közös osztó önmaga
- felbontás: a sorozat két részsorozatra bontása (középen)
- $\boxed{X_1, \dots, X_k} \quad \boxed{X_{k+1}, \dots, X_n}$
- 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.

Az N elemű X tömbben lévő számok közös osztóját kapjuk függvényértékként:

Legnagyobbközösosztó (X, E, U) :

Ha $U-E=0$ **akkor** $L:=X(E)$

különben $K:=(E+U)/2$; $L:=1$

$L1:=$ Legnagyobbközösosztó (X, E, K)

Ha $L1>1$ **akkor** $L2:=$ Legnagyobbközösosztó ($X, K+1, U$)

Ha $L2>1$ **akkor** $L:=\text{luko}(L1, L2)$

Elágazások vége

Legnagyobbközösosztó= L

Függvény vége.

Feladat a Nemes Tihamér OITV-ből és az Informatika OKTV-ből

Kastélyokban különböző számú, különböző méretű téglalap alakú terem van. A kastélyok felújításakor egy kastélyon belül minden terembe egyforma négyzet alakú járólappokat szeretnének lerakni.

Készíts programot, amely beolvassa a kastélyok számát ($1 \leq K \leq 10$), majd a kastélyok termeinek számát ($1 \leq T_i \leq 10$), illetve méreteit ($1 \leq S_{i,j}, H_{i,j} \leq 1\,000\,000\,000$). Ezek alapján minden kastélyra kiszámítja a legnagyobb lehetséges járólappméretet!

Példa

Bemenet

```
K=3
1. kastély
teremszáma: 3
méretek: 10 15 40 40 75 25
2. kastély
teremszáma: 2
méretek: 42 140 70 56
3. kastély
teremszáma: 1
méretek: 154 15
```

Kimenet

```
1. kastély járólappmérete: 5
2. kastély járólappmérete: 14
3. kastély járólappmérete: 1
```

A legnagyobb járólappméret az egyes oldalhosszak legnagyobb közös osztója!

Jelölje $T(i)$ az i -edik kastély termei számát, $S(i, j), H(i, j)$ pedig a j -edik termék szélességét és hosszúságát! Az eredményt az M tömb tartalmazza, azaz az $M(i)$ az i -edik kastély járólappmérete lesz.

Az egyes kastélyok egyes termeihez meg kell határozni a legnagyobb járólappméretet ($m_{i,j} = \text{luko}(S_{i,j}, H_{i,j})$), majd ezek legnagyobb közösosztóját ($M_i = \text{luko}(m_{i,1}, m_{i,2}, \dots, m_{i,T_i})$). Ez lesz az adott kastélyban alkalmazandó járólappméret.

Kastélyok (K, T, S, H, M) :

Ciklus $i=1$ -től K -ig

$M(i) :=$ Járólapp(i, T, S, H)

Ciklus vége

Eljárás vége.

```

Járólap(i,T,S,H) :
  Ciklus j=1-től T(i)-ig
    mm(j) :=lnko(S(i,j),H(i,j))
  Ciklus vége
  Járólap:=Legnagyobbközösosztó(mm,1,T(i))
Eljárás vége.

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

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Haladó
Téma	Számelméleti algoritmusok
Feladat	47. Kastély

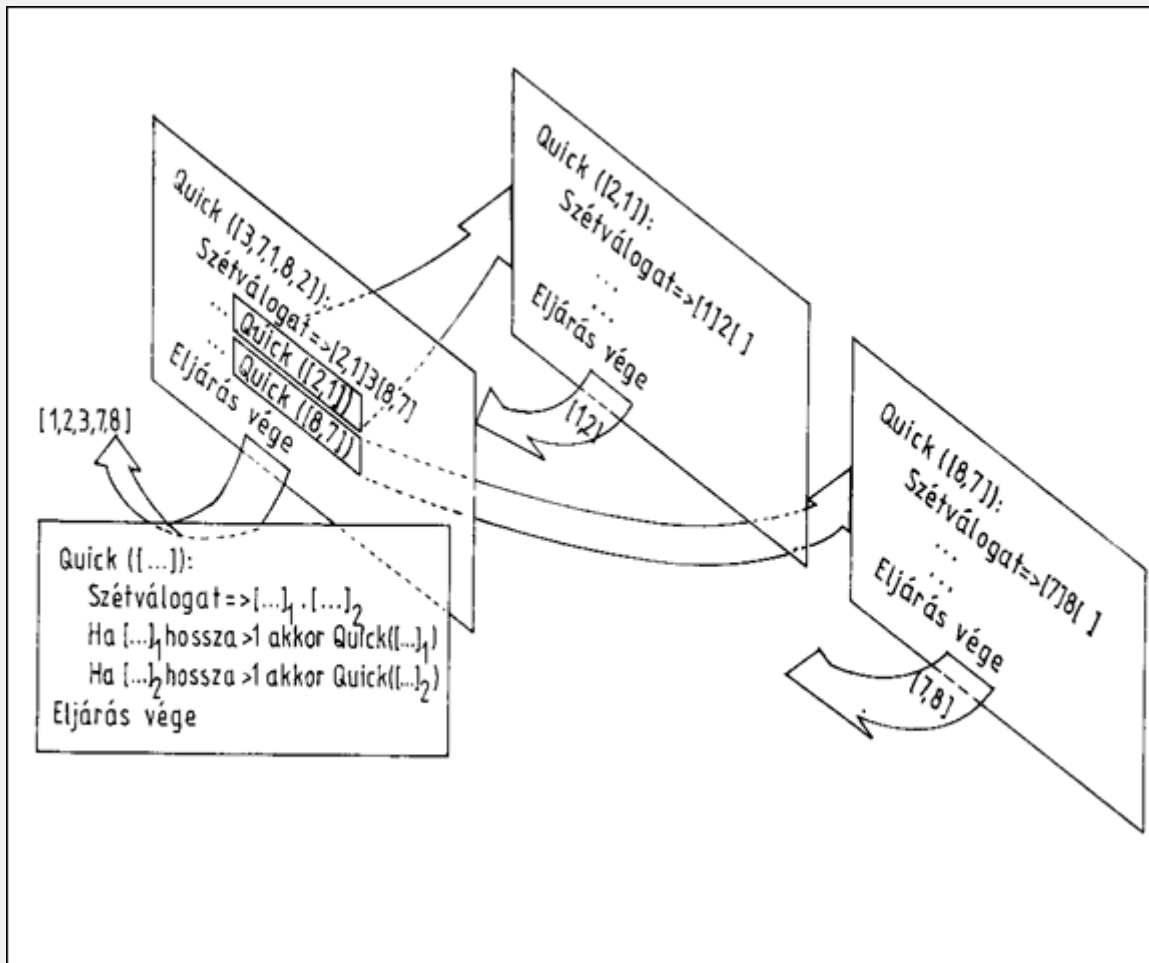
Gyorsrendezés (quicksort)

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: $[X_1, \dots, X_{k-1}] \ X_k \ [X_{k+1}, \dots, X_n]$ szétválogatás (ahol $\forall i,j (1 \leq i \leq k; k \leq j \leq n): X_i \leq X_j$)
- 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 előlről keresünk egy a kivettnél nagyobb elemet, s ha találtunk azt hátra tesszük. 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.
    
```

Ha feltehetjük, hogy a szétválogatásban az elemszámok feleződnek (ez az intervallum első eleme helyett egy ügyes választással megoldható), akkor a rendezés futási ideje $N \cdot \log(N)$ - nagyságrendű lesz.

Feladat a Nemes Tihamér OITV-ből és az Informatika OKTV-ből

N kupacba helyeztek el kavicsokat. KDb darab kavicsot kell elvenni közülük, egyesével. Minden egyes kavics elvétele annyi forintot ér, ahány elemszámú kupacból vesszük el.

Készíts programot, amely beolvassa a kupacok N ($1 \leq N \leq 1\,000\,000$), a kavicsok K ($1 \leq KDb \leq 1\,000$) számát, és az egyes kupacokban lévő kavicsok D_i ($1 \leq D_i \leq 1\,000\,000$) darabszámát; majd megadja, hogy a K kavics elvételével maximum mennyi forintot kereshetünk!

Példa

Bemenet

```
Kupacok száma=3
Kavicsok száma=4
1. kupac kavicsszáma: 6
2. kupac kavicsszáma: 7
3. kupac kavicsszáma: 2
```

Kimenet

```
Maximális nyereség: 24
```

A megoldás gondolatmenete: a legnagyobb méretű kupacokból vegyünk ki kellő számú kavicsot, így jutunk a maximális értékhez! Ehhez a kupacméretek sorozatát csökkenőleg kell rendeznünk. A Gyorsrendezéssel $N \cdot \log_2 N$ hatékonysággal lehet rendezni a kupacdarabszámokat.

Jelöljük $D(i)$ -vel az egyes kupacokban kezdetben lévő kavicsok számát! A fenti algoritmusbeli X tömb helyébe – értelemszerűen – a D tömb lép. Továbbá a rendezés irányát meghatározó relációk megfordulnak, hiszen most nem növekvően, hanem csökkenően kell rendezni.

A megoldás során arra törekszünk, hogy az első i kupac azonos j elemszámú legyen, mert akkor mindegyikből egyet elvéve $i \cdot j$ forintot kereshetünk – de figyelni kell arra, hogy már lehet, hogy nincs szükségünk i számú kavicsra!

```
Kavicsok (N, D, KDb, Ft) :
  Quick(1, N)
  [D tömb csökkenőleg rendezett tömb]
  Ft:=0 [nyereség]
  i:=1; j:=D[1]
  Ciklus amíg i+1 ≤ n és j=D[i+1]
    i:=i+1
  Ciklus vége
  Ciklus amíg kdb > 0
    Ha i > kdb akkor i:=kdb
    Ft:=Ft+i*j; j:=j-1; kdb:=kdb-i
    Ciklus amíg i+1 ≤ n és j=D[i+1]
      i:=i+1
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Kezdő
Téma	Programozási tételek: rendezések
Feladat	29. Kavicsok

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: $[X_1, \dots, X_{k-1}]$ X_k $[X_{k+1}, \dots, X_n]$ szétválogatás (ahol $\forall i, j$ ($1 \leq i \leq k; k \leq j \leq n$): $X_i \leq X_j$)
- 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.

Ha feltehetjük, hogy a felbontásnál az elemszámot megfelezzük, akkor a lépések száma $N+N/2+N/4+\dots < 2*N!$

Nem létező elem keresése

Egy szekvenciális fájlban 1 és *M* (pl. $M=1\,000\,000\,000$) közötti egész számok találhatóak. $1\,000\,000\,000$ 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 végig olvassuk 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: $[1, \dots, K]$ $[K+1, \dots, M]$ 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.
```

Így a megoldásban a számlálást $\log_2 M$ -szer hajtjuk csak végre.

Összefésüléses rendezés (mergesort)

Ha van két rendezett sorozatunk, akkor abból az összefésülés algoritmus 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): $\boxed{X_1, \dots, X_k} \boxed{X_{k+1}, \dots, X_n}$
- 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.
```

Mivel az összefésülés ideje arányos az összefésülendő hosszával, ezért a rendezés futási ideje $N \cdot \log_2 N$ nagyságrendű.

Inverziók száma

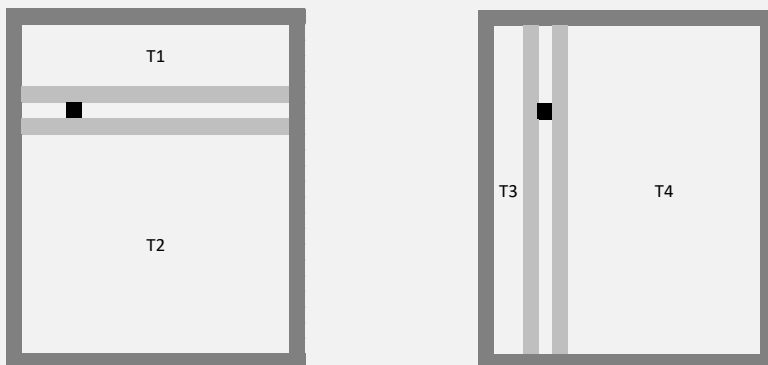
Az inverziók száma egy sorozatban azon (i, j) elempárok száma, ahol $i < j$ és $X(i) > X(j)$. A megoldás nagyon hasonló az összefésüléssel történő rendezéshez. az inverziók száma = az inverziók száma az első félben + az inverziók száma a második félben + az inverziók száma a két fél között (a megoldás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

- leállási feltétel: ha a sorozat legfeljebb 1 elemű, az inverziók száma 0
- felbontás: a sorozat két részsorozatra bontása (középen): $[X_1, \dots, X_k] [X_{k+1}, \dots, X_n]$
- 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 $A \dots B$ (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.

```

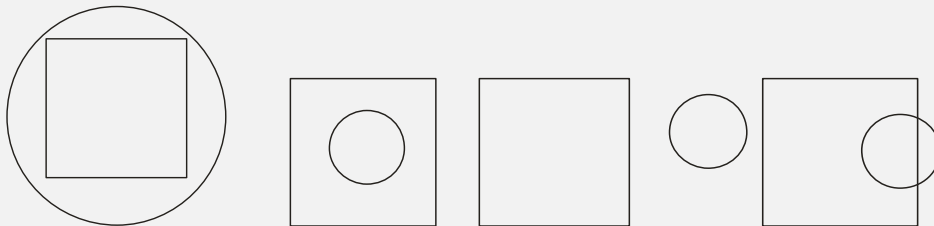
Téglalap(A,B,N,P,C,D,T) :
  Ha belül(A,B,P(N)) akkor
    Ha A.x=P(N).x akkor T:=0
    különben B1.x:=P(N).x-1; B1.y:=B.y
              Téglalap(A,B1,N-1,P,C,D,T)
    Ha B.x=P(N).x akkor T2:=0
    különben A2.x:=P(N).x+1; A2.y:=A.y
              Téglalap(A2,B,N-1,P,C2,D2,T2)
    Ha T2>T akkor T:=T2; C:=C2; 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!)

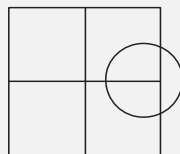
Egy négyzet és egy kör egymáshoz viszonyítva négy különböző helyzetben lehet:

- 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 $E := \text{Esetválasztás}(P, H, Q, R)$

Ha $E = 1$ **akkor** $T := H * H$

különben **ha** $E = 2$ **akkor** $T := R * R * \pi$

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 eljárásán kívülre tehetjük.

Karatsuba szorzási algoritmus

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: $X * Y = (X_1 * 2^N + X_2) * (Y_1 * 2^N + Y_2) = X_1 * Y_1 * 2^{2N} + X_1 * Y_2 * 2^N + X_2 * Y_1 * 2^N + X_2 * Y_2 = X_1 * Y_1 * 2^{2N} + (X_1 * Y_2 + X_2 * Y_1) * 2^N + X_2 * Y_2$.

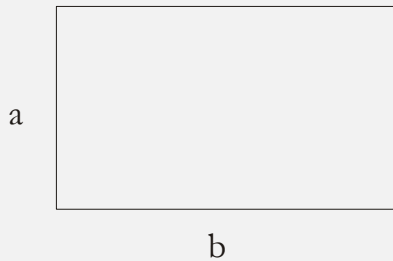
Azaz minden szorzás visszavezethető feleakkora számok 4 szorzására, valamint eltolásokra. Ügyes átrendezéssel azonban három szorzás is elég lehet:

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

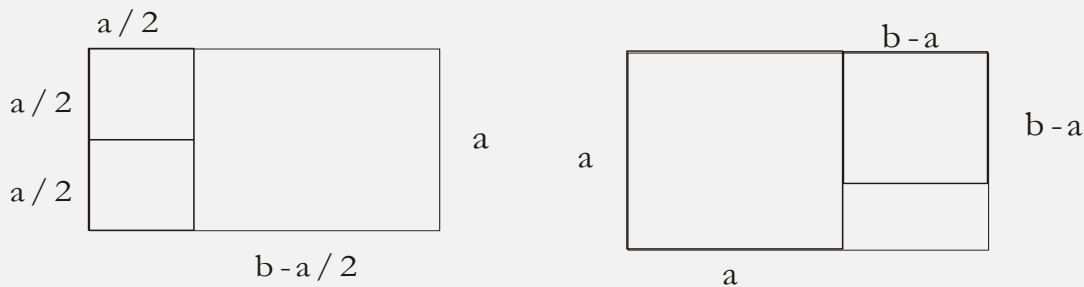
$X_1 * Y_1 + X_1 * Y_2 + X_2 * Y_1 + X_2 * Y_2$, ha ebből levonjuk $X_1 * Y_1$ és $X_2 * Y_2$ értékét – amit már kiszámoltunk – akkor éppen a fenti képletben szereplő zárójeles kifejezést kapjuk.

Téglalap felbontás

Adott egy $a \times b$ méretű téglalap ($b-a < a < b$).



Egy ilyen alakzat kétféleképpen bontható két négyzetre és egy téglalagra:



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 téglalagra
- uralkodás: a téglalapokra a minimális lépésszám kiszámolása (rekurzívan)
- összevonás: a két lépésszám minimumának meghatározása.

Felbontás (A, B, L) :

Ha $B-A \geq A$ **vagy** $A \geq B$ **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.