Algoritmizálás

Horváth Gyula
Szegedi Tudományegyetem
Természettudományi és Informatikai Kar
horvath@inf.u-szeged.hu

8. Interaktív feladatok

8.1. Feladat: Számjáték (IOI'96)

Adott az alábbi kétszemélyes játék. A játéktábla pozitív egész számok sorozata. A két játékos felváltva lép. Egy lépés azt jelenti, hogy a játékos sorozat bal vagy jobb végéről kiválaszt egy számot. A kiválasztott számot törlik a tábláról. A játék akkor ér véget, ha a számok elfogytak. Az első játékos nyer, ha az általa választott számok összege legalább annyi, mint a második játékos által választottak összege. A második játékos a lehető legjobban játszik. A játékot az első játékos kezdi. Ha kezdetben a táblán levő számok száma páros, akkor az első játékosnak van nyerő stratégiája.
Írjunk olyan programot, amely az első játékos szerepét játssza és megnyeri játékot! A második játékos lépéseit egy már adott számítógépes program szolgáltatja. A két játékos a rendelkezésedre bocsátott `Play` modul három eljárásán keresztül kommunikál egymással.

StartGame Az első játékos a játszmát a paraméter nélküli StartGame eljárás végrehajtásával indítja.

MyMove Ha az első játékos a bal oldalról választ számot, akkor a `MyMove('L')` eljárást hívja. Hasonlóképpen a `MyMove('R')` hívással közli a második játékossal, hogy a jobb oldalról választott.

YourMove A második játékos (tehát a gép) azonnal lép. Az első játékos a lépést a `YourMove(C)` utasítással tudhatja meg, ahol C egy karakter típusú változó. (C/C++ nyelven `YourMove(&C)`). A C változó értéke 'L' vagy 'R' lesz attól függően, hogy a második játékos a bal vagy a jobb oldalról választott.

Bemenet

Az `input.txt` fájl első sora a kezdőtábla n méretét (a számok darabszámát) tartalmazza. n páros és $2 <= n <= 100$. A második sor n számot tartalmaz, a játék kezdetén a táblán lévő számokat. A táblán 200-nál nagyobb szám nem szerepel.

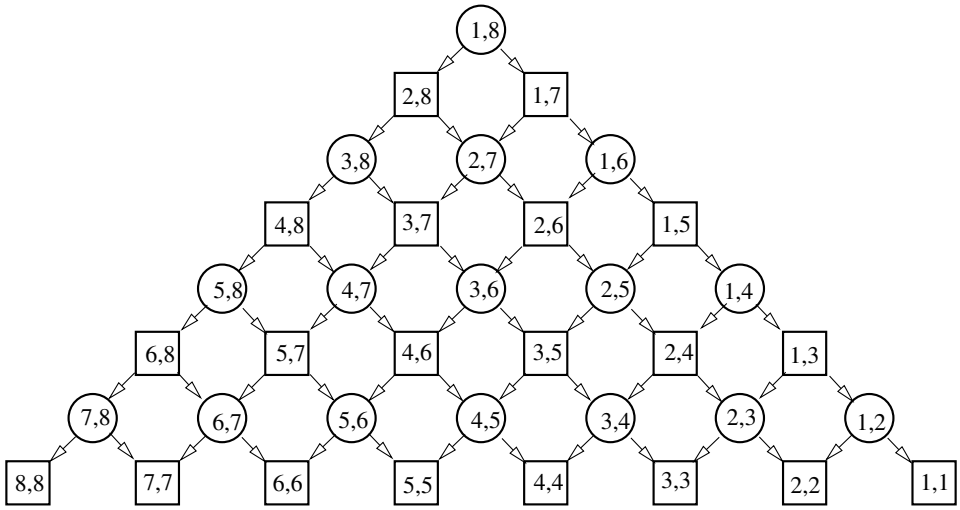
Kimenet

Ha a játék véget ért, akkor a programod írja ki a végeredményt az `OUTPUT.TXT` fájlba! A fájl első sorában két szám legyen! Az első szám az első játékos által választott számok összegével, a második szám a második játékos által választott számok összegével egyezzen meg! A programodnak a játékot le kell játszania és az output a lejátszott játék eredményét kell tartalmazza.

Példa bemenet és kimenet

INPUT.TXT	OUTPUT.TXT
6 4 7 2 9 5 2	18 11

Megoldás
Jelölje $\langle a_1, \ldots, a_n \rangle$ a kezdeti játékállást. Minden lehetséges játékállást egyértelműen meghatározza az, hogy mely számok vannak még a táblán. Tehát minden játékállás azonosítható (i, j) számpárral, ami azt jelenti, hogy a táblán az $\langle a_i, \ldots, a_j \rangle$ számsorozat van. Mivel n páros szám, így minden esetben, amikor az első játékos lép, vagy i páros és j páratlan, vagy fordítva. Tehát az első játékos kényszerítheti a második játékost, hogy az mindig vagy csak páros, vagy csak páratlan indexű elemét válassza a számsorozatnak. Tehát ha a páros indexűek összege nagyobb, vagy egyenlő, mint a páratlanok összege, akkor az első játékos mindig páratlan indexűt választ, egyébként mindig párosat.
Érdekesebb a játék, ha az a cél, hogy az első játékos a lehető legtöbbet szerezze meg, feltéve, hogy erre törekszik a második játékos is. Ábrázoljuk a játékállásokat gráffal.

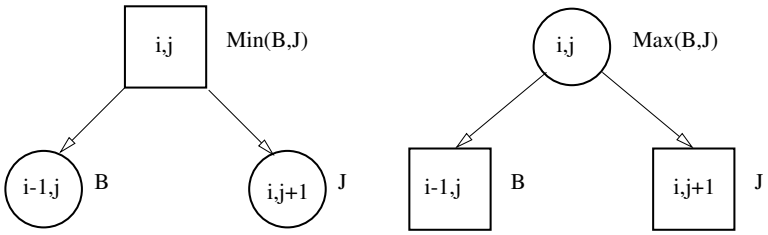


1. ábra. A játékállások gráfja $n = 8$ esetén. Körrel jelölt állásból ($i + j$ páratlan) az első, négyzettel jelölt állásból ($i + j$ páros) a második játékos lép.

Definiáljuk minden (i, j) játékállásra azt a maximális pontszámot, amit az első játékos elérhet ebből a játékállásból indulva. Jelölje ezt az értéket $Opt(i, j)$. $Opt(i, j)$ a következő rekurzív összefüggés számítható.

$$Opt(i,j) = \begin{cases} 0 & \text{ha } i = j \\ \max(a_i + Opt(i+1,j), a_j + Opt(i,j-1)) & \text{ha } i < j \text{ és } i+j \text{ páratlan} \\ \min(Opt(i+1,j), Opt(i,j-1)) & \text{ha } i < j \text{ és } i+j \text{ páros} \end{cases}$$

Tehát alkalmazható a dinamikus programozás módszere, vagyis az $Opt(i, j)$ értékeket a játék megkezdése előtt kiszámítjuk.



2. ábra. Mini-max szabály.

Tároljuk minden (i, j) játékállásra a `Lep[i,j]` tömbelemben az optimális lépést, tehát az 'L' karaktert, ha a képletben $a_i + Opt(i+1, j) > a_j + Opt(i, j-1)$, mert ekkor balról kell elvenni, egyébként pedig az 'R' karaktert, mert ekkor jobbról kell elvenni.

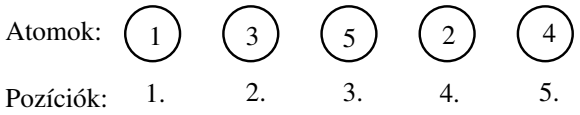
```
1  program Jatek;
2  uses Play; {a masodik játékost megvalósító modul}
3  const
4    MaxN=100;
5  var
6    InpF,OutF: Text;
7    A:array[1..MaxN] Of Word;{ a táblán lévő számok sorozata}
8    N: Word;                 { a tábla mérete}
9    Opt:array[1..MaxN,1..MaxN] of word;
10   Lt:array[1..MaxN,1..MaxN] of Char; {az 1. játékos optimális lépései}
11
12 procedure Beolvas;
13 var i:Word;
14 begin
15   Assign(InpF,'input.txt'); Reset(InpF);
```

```
16    ReadLn(InpF,N);
17    for i:=1 to N do
18      Read(InpF,A[i]);
19    Close(InpF);
20  end;

21  procedure Elofeldolgoz;
22  var i,j:Word;
23    Pont,PontBal,PontJobb:Word;
24  begin
25    for j:=1 to N do begin
26      Opt[j,j]:=0;
27      for i:=j-1 DownTo 1 do begin
28        if Odd(j-i+1) then begin{2. játékos lép}
29          if Opt[i+1,j]<Opt[i,j-1] then
30            Opt[i,j]:=Opt[i+1,j]
31          else
32            Opt[i,j]:=Opt[i,j-1]
33        end else begin{1. játékos lép}
34          PontBal:=A[i]+Opt[i+1,j];
35          PontJobb:=A[j]+Opt[i,j-1];
36          if PontBal>PontJobb then begin
37            Opt[i,j]:=PontBal;   Lt[i,j]:='L'
38          end else begin
39            Opt[i,j]:=PontJobb; Lt[i,j]:='R'
40          end
41        end;
42      end{for i};
43    end{for j};
44  end {Elofeldolgoz};

45  procedure Jatszas;
46  var
47    Bal,Jobb:Word;{az aktuális játékállás: A[Bal..Jobb]}
48    L1,L2: Char;  {a két játékos aktuális lépése}
49  begin
50    Bal:=1; Jobb:=N;              {a kezdő játékállás beállítása}
51    while Bal<=Jobb do begin      {amíg nem üres a tábla}
52      MyMove(Lt[Bal, Jobb]);      {az én lépésem}
53      if Lt[Bal, Jobb]='L' then {a játékállás aktualizálása}
54        Inc(Bal)
55      else
56        Dec(Jobb);
57      L2:=YourMove;               {az ellenfél lépése}
58      if L2='L' then              {a játékállás aktualizálása}
59        Inc(Bal)
60      else
61        Dec(Jobb);
62    end{while};
63  end{Jatszas};
64  begin
65    Beolvas;
66    Elofeldolgoz;
67    StartGame;
68    Jatszas;
69  end.
```

Kitalálós feladatok

8.2. Feladat: Atomlánc (CEOI'2001)

Kutatók egy speciális molekulát vizsgálnak. Tudják, hogy a molekula n különböző atomot tartalmaz, amelyek egy lineáris láncot alkotnak. A kutatóknak van egy olyan mérőműszerük, amellyel meg tudják mérni a molekula két adott atomjának a távolságát. A műszer által mért távolság a két atom pozíció különbségének abszolút értéke. Az elvégezhető mérések azonban korlátozottak, egyetlen atom sem szerepelhet négynél több mérésben, mert ez tönkretenné a molekulát.
Olyan programot kell írni, amely meghatározza a molekula szerkezetét, azaz minden atom pozícióját a molekulában!



3. ábra. Atomlánc példa

A mérőműszer használatát a Meter könyvtár három művelete biztosítja:
Size Egyszer kell hívni a program elején, az atomok n számát adja.

Span Két atom sorszámát kell argumentumként megadni, a visszaadott érték a két atom távolsága.

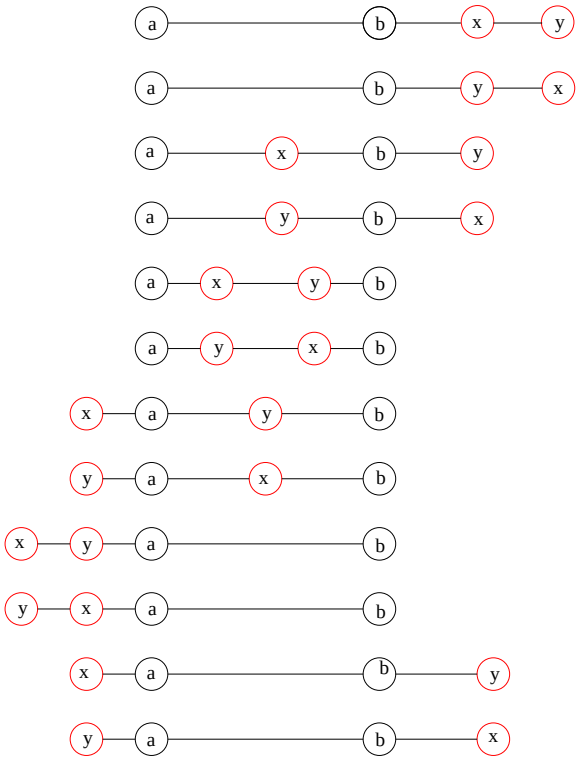
Answer A program végén kell hívni, a kiszámított eredmény közléséhez. Két argumentuma van, i és x, ami azt jelenti, hogy a molekulában az i-edik pozíción a x sorszámú atom van. Minden i-re $(1 \leq i \leq n)$ pontosan egyszer kell hívni, tetszőleges sorrendben. A megoldás tükörkép erejéig egyértelmű, a két megoldás közül bármelyiket meg lehet adni.

A `Meter` könyvtár két szöveges állományt készít: `chain.out` és `chain.log`. A `chain.out` két sort tartalmaz, az első sor a program által egy atomra végrehajtott legtöbb `Span` művelet számát tartalmazza. Ez a szám legfeljebb 5 lehet. A második sor az Answer műveletekkel közölt atom sorszámok sorozatát tartalmazza. A program és könyvtár közötti dialógust `chain.log` tartalmazza.
Utasítások Pascal programozóknak:
A `uses meter;` import utasítás szerepeljen a program első sorában.
Utasítások C/C++ programozóknak:
A `#include` "meter.h " import utasítás szerepeljen a program első sorában. Készítsen egy `chain.gpr` projekt állományt a feladatköyvtárban és adja hozza a projekthez a `chain.c (chain.cpp)` és `meter.o` állományokat és compile/make paranccsal végezze a fordítást.
HASZNÁLAT Készíteni kell egy `meter.in` szöveges állományt, amely két sort tartalmazzon. Az első sorban legyen az atomok n száma. A második sor pontosan n különböző számot tartalmazzon egy-egy szóközzel elválasztva, az atomok sorszámait. Minden szám 1 és n közötti egész szám legyen.

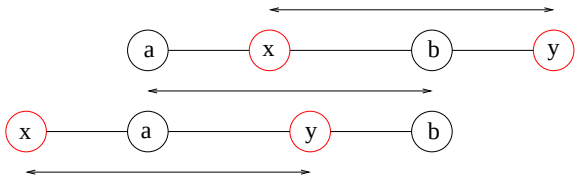
Példa bemenet és kimenet

`meter.in`
5
1 3 5 2 4
Eljárás hívások, amelyek egy megoldását adják a példa bemenetnek:
1. Size (Pascal-ban) vagy Size() (C/C++-ban) 5 értéket ad
2. Span(1,3) 1-et ad
3. Span(2,5) 1-et ad
4. Span(3,5) 1-et ad
5. Span(1,4) 4-et ad
6. Answer(1,1) Answer(5,4) Answer(3,5) Answer(4,2) Answer(2,3)
FELTÉTELEK

- Az atomok n számára: $5 \leq n \leq 10000$
- Ha bármely atom négynél többször szerepel Span műveletben, akkor a program megszakítását eredményezi.
- A megoldás program nem olvashat és nem írhat semmilyen állományt.



5. ábra. x és y lehetséges elhelyezkedése $a-b$-hez képest.



6. ábra. **Vigyázat!** Ha $Tav(a,b) = Tav(x,y)$, akkor nem egyértelmű. A két lehetséges esetet mutatja az ábra.
De ha van három olyan a, b és c atomunk, amelyek pozícióját ismerjük, és mindegyik legfeljebb 2 kérdésben szerepelt, akkor ezek közül kettő távolsága biztosan nem egyenlő $Tav(x,y)$-al. Ez biztosítható kezdetben úgy, hogy meghatározzuk az 1. 2. és 3. atom pozícióját.

- Az atomok pozícióira: $1 \leq i \leq n$.

```
1  program Chain;
2  uses Meter;
3  const
4    MaxN=10000;                        {az atomok max. száma}
5  var
6    N:Integer;                         {az atomok száma}
7    S:array[0..MaxN] of Integer;       {a megoldás}
8    i:Integer;
9
10 procedure Szamit; {Global: N, S }
11   var
12     Poz:array[0..MaxN] of -MaxN..MaxN;{relatív atom pozíciók 1-hez képest}
13     a,b,c,x,y:Integer;                {atom címkék}
14     Dab,Dxy,Dax,Dbx,Dby:Integer;      {távolságok}
15     i,k,min,z:Integer;
16
17 begin{Szamit};
18   a:=1; b:=2; c:=3;  {3 atom választása}
19   Dab:=Span(a,b); Dax:=Span(a,c); Dbx:=Span(b,c);
20   Poz[1]:=0; Poz[2]:=Dab;

21   if (Dax=Dab+Dbx) or (Dab=Dax+Dbx) then {c  a->b relatív pozíciójának kiszámítása}
22     Poz[c]:=Poz[a]+Dax
23   else if (Dbx=Dax+Dab) then
24     Poz[c]:=Poz[a]-Dax;
25   i:=3;
26   while i+2<=N do begin
27   {a,b és c 3-szor szerepelt kérdésben és rel. pozíciójuk ismert}
28     x:=i+1; y:=i+2;          {a következő 2 atom: x, y}
29     Inc(i,2);
30     Dxy:=Span(x,y);          { x és y távolságának lekérdezése}
31     {válasszunk 2 atomot [a,b,c] közül, hogy távolságuk különböző
32     legyen Dxy-től.}
33     if Dxy<>Abs(Poz[b]-Poz[c]) then begin          {b--c<>x--y}
34       z:=a; a:=c; c:=z;
35     end else if Dxy<>Abs(Poz[a]-Poz[c]) then begin{a--c<>x--y}
36       z:=b; b:=c; c:=z;
37     end;  {else a--b<>x--y }
38     Dab:=Abs(Poz[a]-Poz[b]);
39     if Poz[b]<Poz[a] then begin{ Poz[a]<Poz[b] biztosítása}
40       z:=a; a:=b; b:=z;
41     end;

42     {Poz[a]<Poz[b], Dab<>Dxy}
43     Dax:=Span(a,x);
44     Dby:=Span(b,y);
45     { x és y reltív pozíciójának meghatározása}
46     if (Dax=Dab+Dby+Dxy) Or                  {a--b--y--x}
47        (Dax+Dxy=Dab+Dby) then begin          {a--b--x--y}
48       Poz[x]:=Poz[a]+Dax;                    {a--x--b--y}
49       Poz[y]:=Poz[b]+Dby;
50     end else if (Dab=Dax+Dxy+Dby) Or         {a--x--y--b}
51        (Dab+Dxy=Dax+Dby) then begin          {a--y--b--x}
52       Poz[x]:=Poz[a]+Dax;                    {a--y--x--b}
53       Poz[y]:=Poz[b]-Dby;                    {y--a--b--x}
54                                              {y--a--x--b}
```

```
55      end else if (Dxy=Dab+Dax+Dby) then begin{x--a--b--y}
56        Poz[x]:=Poz[a]-Dax;
57        Poz[y]:=Poz[b]+Dby;
58      end else if (Dby=Dxy+Dax+Dab) Or          {y--x--a--b}
59                  (Dax+Dab=Dby+Dxy) then begin{x--y--a--b}
60        Poz[x]:=Poz[a]-Dax;                     {x--a--y--b}
61        Poz[y]:=Poz[b]-Dby;
62      end{if};
63      a:=x; b:=y;                    { a, b helyett x és y }
64    end{while};

65    if Not Odd(N) then begin{az utolsó elem feldolgozása, ha N páros}
66      if Poz[b]<Poz[a] then begin
67        z:=a; a:=b; b:=z;
68      end;
69      Dab:=Abs(Poz[a]-Poz[b]);
70      Dax:=Span(a,N);
71      Dbx:=Span(b,N);;
72      if Dax=Dab+Dbx then
73        Poz[N]:=Poz[b]+Dbx
74      else if Dab=Dax+Dbx then
75        Poz[N]:=Poz[a]+Dax
76      else
77        Poz[N]:=Poz[a]-Dax
78    end;
79    min:=Poz[1];
80    for i:=2 to N do
81    if Poz[i]<min then min:=Poz[i];
82    min:=-min+1;
83    for i:=1 to N do      {a relatív pozíciók 1..N tartományba léptetése}
84      S[Poz[i]+min-1]:=i;
85  end{Szamit};

86  begin{program}
87    N:=Size;
88    Szamit;
89    for i:=0 to N-1 do
90      Answer(i+1,S[i]);
91  end.
```

8.3. Feladat: Többségi elem kiválasztása (CEOI'2001)

Iskolád tanulói két csoportba tartoznak. Tudjuk, hogy az egyik csoportban többen vannak, mint a másikban, ezt nevezzük többségi csoportnak. Ki kell választani egy tanulót, aki a többségi csoporthoz tartozik. Ehhez egyetlen műveletet használhatunk, nevezetesen két tanulótól megkérdezhetjük, hogy ugyanabba a csoportba tartoznak-e.
Olyan programot kell írni, amelyik a lehető legkevesebb kérdéssel meghatároz egy többségi csoporthoz tartozó tanulót. A tanulókat sorszámukkal azonosítjuk.

KÖNYVTÁRI MŰVELETEK

A feladat megoldásához három könyvtári művelet van.

Size A tanulók n számát adja. Ezt kell először hívni.

Member Két tanuló sorszámát kell argumentumként megadni, és a függvényeljárás 1 értéket ad, ha a két tanuló ugyanazon csoport eleme, egyébként 0-át.

Answer Ezzel a művelettel kell közölni a kiválasztott, többségi csoportba tartozó tanuló sorszámát.

A programod és a könyvtári modul közötti párbeszédet a `select.out` szöveges állományban rögzítik. Ez a nap állomány a programod által közölt megoldást is tartalmazza, továbbá azt is, hogy az helyes-e. A megoldást csak akkor fogadják el, ha a

tanulók bármely olyan diszjunkt A és B részhalmazára, amely kompatibilis az általad feltett kérdésekkel, a közölt megoldás a nagyobb elemszámú részhalmazban van.
A válaszadó arra kényszerít, hogy szükséges számú kérdést tegyél fel.
Pascal program esetén
`uses query;` import direktívát kell használni.
C/C++ program esetén
`#include` "query.h " direktívát kell használni.
GYAKORLÁS
A könyvtári modul úgy használható, hogy a `select.in` szöveges állomány első és egyetlen sorába a tanulók n számát kell írni, ami páratlan szám kell legyen!

Példa bemenet és kimenet

select.in	select.out
7	Size=7
	Member(1,2)=0
	Member(3,4)=1
	Member(5,6)=1
	Member(4,6)=0
	Your Answer: 7, Correct
	Majority Group:
	2 5..7
	Non-majority Group:
	1 3 4
	Number of Queries: 4
	Full Possible Score: 3
	Your Score: 3

Például, az 1 válasz nem elfogadható, mert minden feltett kérdésre a $\{2,5,6,7\}$ és $\{1,3,4\}$ halmazok esetén a MEMBER függvény ugyanazt eredményezné, de 1 nem eleme a $\{2,5,6,7\}$ többségi csoportnak. Az $a..b$ jelölés az a és b közötti egész számok halmazát jelenti. **FELTÉTELEK**

- A tanulók n számára $5 \leq n \leq 30000$, n páratlan teljesül.
- A programod nem olvashat és nem írhat semmilyen filet.
- A tanulók azonosítói: $1 \leq i \leq n$
- FreePascal könyvtári modulok filenevei: `query.ppw`, `query.ow`.
- A könyvtári műveletek Pascal deklarációi:
```
function Size:  integer;
function Member(x, y:  integer):  integer;
procedure Answer(x:  integer);
```
- C/C++ könyvtári modulok filenevei: `query.h` , `query.o`
- C/C++ deklarációk:
```
int Size(void);
int Member(int x, int y);
void Answer(int x);
```

PONTOZÁS
Helyes válasz esetén a kapott pontszám: $\max(0, n-k)$, ha a programod k MEMBER műveletet hajtott végre.
Megoldás
Jelölje $H = \{1,\ldots,n\}$ a tanulók halmazát. Azt mondjuk, hogy egy $A \subseteq H$ **homogén** részhalmaz, ha A minden eleme ugyanabba a csoportba tartozik, azaz ha $\forall x,y \in A$,$member(x,y) = 1$. Azt mondjuk, hogy $U,V \subseteq H$ **ellentétes** részhalmazok, ha U minden eleme az egyik, V minden eleme a másik csoportba tartozik, azaz ha $\forall x \in U$ $\forall y \in V$,$member(x,y) = 0$.

Vegyük észre, hogy ha $member(x,y) = 0$, akkor x és y elhagyható a halmazból, mert biztosan marad még többségi csoportba tartozó elem. Általánosan, ha $U,V \subseteq H$ homogén részhalmazok és elemszámuk megegyezik, továbbá valamely $x \in U$ és $y \in V$

elemekre $member(x,y) = 1$ -et kapunk, akkor U és V törölhető H-ból.
Bármely kérdéssorozat által nyert ismeret ábrázolható egy olyan halmazzal, amelynek az elemei diszjunkt halmazpárok. Pontosabban, az alábbi feltételeket kielégítő halmazzal.

$$I = \{ (U_1,V_1),\ldots,(U_m,V_m) \} \tag{1}$$

$$\text{az } U_i,V_j \text{ halmazok páronként diszjunktak } 1 \le i,j \le m \tag{2}$$

$$U_i \text{ és } V_i \text{ homogén részhalmaz }, 1 \le i \le m \tag{3}$$

$$U_i \text{ az } V_i \text{ ellentétesek } 1 \le i \le m \tag{4}$$

$$\bigcup_{i=1}^{m} (U_i \cup V_i) = \{ 1,\ldots,n \} \tag{5}$$

$$\tag{6}$$

A kezdeti helyzetet, amikor nincs semmi ismeretünk, az

$$I = \{ (\{ 1 \},\emptyset),\ldots,(\{ n \},\emptyset) \} \tag{7}$$

halmaz ábrázolja. Tegyük fel, hogy az eddig elvégzett kérdésekkel nyert információt az (4) halmaz ábrázolja, és a $Member(x,y)$ kérdést tesszük fel. Mivel az I-beli részhalmazok páronként diszjunktak, pontosan egy olyan i és pontosan egy olyan j index van, hogy

$$x \in U_i \cup V_i \text{ és } y \in U_j \cup V_j$$

Ha $i = j$, akkor a kérdés redundáns, azaz a kérdésre a válasz megadható a korábbi kérdésekre kapott válaszok alapján, nevezetesen, a válasz $Igaz$, ha $x \in U_i$ és $y \in U_i$ vagy $x \in V_i$ és $y \in V_i$, egyébként a válasz 0. Ekkor nem jutunk új ismerethez.
Ha $i \neq j$, akkor új ismerethez jutunk, amit az

$$I' = I - \{ (U_i,V_i),(U_j,V_j) \} \cup \{ (U,V) \} \tag{8}$$

halmazzal ábrázolhatunk, ahol az (U,V) párt az alábbiak szerint kapjuk.
Ha $Member(x,y) = Igaz$, akkor

$$(U,V) = \begin{cases} (U_i \cup U_j, V_i \cup V_j) \text{ ha} & (x,y \in U_i \cup U_j) \vee (x,y \in V_i \cup V_j) \\ (U_i \cup V_j, U_j \cup V_i) \text{ ha} & (x,y \in U_i \cup V_j) \vee (x,y \in U_j \cup V_i) \end{cases}$$

Ha $Member(x,y) = 0$, akkor

$$(U,V) = \begin{cases} (U_i \cup U_j, V_i \cup V_j) \text{ ha} & (x,y \in U_i \cup V_j) \vee (x,y \in U_j \cup V_i) \\ (U_i \cup V_j, U_j \cup V_i) \text{ ha} & (x,y \in U_i \cup U_j) \vee (x,y \in V_i \cup V_j) \end{cases}$$

Nyilvánvaló, hogy bármely x akkor és csak akkor fogadható el a feladat helyes megoldásának, ha az elvégzett kérdésekhez tartozó I ismeretre teljesül, hogy $x \in U_i \cup V_i$ esetén, ha $x \in U_i$ akkor $|U_i| > |V_i|$, illetve ha $x \in V_i$ akkor $|V_i| > |U_i|$, és

$$\max(|U_i|,|V_i|) + \sum_{\substack{j=1 \\ j \neq i}}^{n} \min(\left|U_j\right|,\left|V_j\right|) > \min(|U_i|,|V_i|) + \sum_{\substack{j=1 \\ j \neq i}}^{n} \max(\left|U_j\right|,\left|V_j\right|) \tag{9}$$

Ha az I ismeretre teljesül a (13) egyenlőtlenség valamely i indexre, akkor azt mondjuk, hogy I **biztos** ismeret. Ekkor az U_i és V_i halmazok közül a nagyobbik elemszámú halmaz mindegyik eleme biztosan a többségi csoportba tartozik.
A (13) feltétel ekvivalens a (14) feltétellel.

$$||U_i| - |V_i|| > \sum_{j=1, j \neq i}^{n} \left|\left|U_j\right| - \left|V_j\right|\right| \tag{10}$$

$I := \{\{1\},\ldots,\{n\}\}$;
x:=0;
while $|Max(I)| \leq n/2$ **begin**
  $x := x+1$;
  $y := x+1$;
  if $Member(x,y)$ **then begin**
    $U := \{x,y\}$;
    while van olyan $V \in I$, hogy $|U| = |V|$ **do begin**
      $y := V$ egy tetszőleges eleme;
      $I := I - \{V\}$;
      if Member(x,y) **then**
        $U := U \cup V$;
      else begin
        $n := n - 2|U|$;
        $U := \emptyset$;
        Break;
      end
    end;
    if $U \neq \emptyset$ **then**
      $I := I \cup \{U\}$
  end else
    $n := n-2$;
end;
$t := Max(I)$ egy eleme;

Az algoritmus legfeljebb

$$n - b(n) \tag{11}$$

kérdést tesz fel, ahol $b(n)$ az n szám kettes számrendszerbeli felírásában az egyes bitek száma, azaz

$$b(n) = \sum_{i=1}^{k} b_i \tag{12}$$

ha

$$n = \sum_{i=1}^{k} b_i 2^i (b_k \neq 0) \tag{13}$$

Megvalósítás

```
1  program Select;
2  uses Query;
3  const
4    MaxN=30000;                    {max. tanulószám}
5    MaxK=20;                       {MaxN<=2^MaxK}
6  var
7    N:1..MaxN;                     {atanulók száma}
8    M:0..MaxN;                     {a redukált halmaz elemszáma}
9    Fel:Longint;                   {a többségi csoport elemszáma}
10   B:array[0..MaxK] Of Boolean;   {2^k elemű részcsoportok}
11   R:array[0..MaxK] Of 0..MaxN;   {a részcsoportok egy reprezentánsa}
12   Pow2:array[0..MaxK] Of Longint;{Pow2[k]=2^k}
13   L:Word;                        {a legnagyobb részcsoport elemszáma 2^L}
14   i,k:Integer;
15 begin
16   Pow2[0]:=1;
17   for k:=1 to MaxK do            {2 hatványok kiszámítása}
```

```
18      Pow2[k]:=Pow2[k-1] Shl 1;
19    N:=Size;                          {a tanulók számának lekérdezése}
20    M:=N;
21     Fel:=M div 2 +1;
22    L:=0;   i:=0;

23     while i<N do begin                {M elemű halmaz többségi elemének keresése}
24       k:=0; B[0]:=True;
25       Inc(i); R[0]:=i;
26       Inc(i);                        {a két következő elem: i és i+1}
27       if i>N then Break;             {nincs több}
28       while B[k] do begin            {van két 2^k elemszámú részcsoport}
29         if Member(R[k],i)=1 then begin{i-t és R[k]-t tartalmazó két 2^k elememű}
30           B[k]:=False;               {részcsoport egyesítése}
31           Inc(k);                    {2^k+1 elemű lesz az új részcsoport}
32           if k>L then L:=k;          {új legnagyobb részcsoprt?}
33         end else begin               {nem azonos csoportba tartozó részcsoportok}
34           Dec(M, Pow2[k+1]);         {M:=M-2^(k+1)}
35           Dec(Fel, Pow2[k]);         {Fel:=Fel-2^k}
36           B[k]:=False;               {töröljük a részcsoportot}
37           if k=L then                {L aktualizálása}
38             while (L>0)And Not B[L] do
39               Dec(L);
40           k:=-1;
41           Break;
42         end;
43       end{while B[k]};

44       if k>=0 then begin
45         B[k]:=True;                  {új 2^k elemszámú részcsopportot kaptunk}
46         R[k]:=i;                     {i az új részcsoport reprezentánsa}
47       end;
48
49       if (L>0)And(Pow2[L]>=Fel) then{a legnagyobb részcsoport a többségi?}
50         Break;
51     end{while i<N};
52
53     Answer(R[L]);              {a legnagyobb részcsoport reprezentánsa a megoldás}
54   end.
```

8.4. Feladat: Median (IOI'2000)

Egy űrkísérletben n tárgyat használunk, melyeket 1-től n-ig számozunk, ahol n páratlan. Minden tárgy különböző súlyú (természetes számok), de magukat a súlyokat nem ismerjük. Minden y súlyra igaz, hogy $1 \leq y \leq n$. Mediánnak nevezzük azt a tárgyat, amelyiknél ugyanannyi könnyebb, mint nehezebb tárgy van. Írj programot, amely meghatározza a mediánt! A tárgyak súlyát olyan eszközzel hasonlíthatjuk össze, amely három tárgy közül megadja a mediánt.

Könyvtár

A device nevű könyvtárból az alábbi három művelet használható:

GetN egyszer kell meghívni, a programod legelején; az argumentum nélküli függvényhívás eredménye az n értéke.

Med3 három különböző tárgy sorszámával kell hívni, függvényértéke e három sorszám közül a mediánjuk sorszáma.

Answer egyszer kell meghívni, a programod végén; argumentumként az N tárgy mediánjának a sorszámát kell megadnod. Ez a hívás le is állítja a programodat.

A device könyvtár függvényei két szöveges állományt hoznak létre `MEDIAN.OUT` és `MEDIAN.LOG` néven. A `MEDIAN.OUT` első sorában egy egész szám lesz, az, amit az ANSWER eljárásnak adtál át. A második sorban a MED3 hívások száma lesz. A programod és a könyvtár közötti párbeszédet a `MEDIAN.LOG` tartalmazza.

Pascal programozóknak:
programodba írd be a következő sort: uses device;
Kipróbálás

Programod kipróbálásához készíts `DEVICE.IN` néven olyan állományt, amely két sorból áll. Az elsőbe a tárgyak számát (n) kell írni. A második sor a tárgyak súlyát (1 és n közötti különböző egész számok) tartalmazza, ahol az i-edik érték az i-edik tárgy súlya.
Kikötések

- $5 \leq n \leq 1499$ és n páratlan.
- Minden i sorszámra igaz: $1 \leq i \leq n$.
- Minden y súlyra igaz: $1 \leq yn$ és minden súly különböző.
- A Pascal könyvtár neve: device.tpu
- A Pascal függvények és eljárás deklarációja:
  function GetN: integer;
  function Med3(x,y,z:integer):integer;
  procedure Answer(m:integer);
- Futtatásonként a MED3 legfeljebb 7777-szer hívható.
- Programod nem olvashat és nem írhat egyetlen állományt sem.

Megoldások

Alapelv: ismételten határozzuk meg és távolítsuk el a két szélső elemet.
A $H \subseteq \{1,\ldots,n\}$ halmaznak $a,b \in H$ két szélső eleme, ha minden $x \in H$ elemre, ha $x \neq a$ és $x \neq b$

$$Med3(a,x,b) = x$$

```
1  program Median1; {Hagymahámozó algoritmus}
2  uses Device;
3  const
4    MaxN=3000;
5  type
6    Node=1..MaxN;
7  var
8    N:Node;
9    M:Integer;
10 function Compute:Integer;
11 var
12   S:array[1..MaxN] Of 0..MaxN;
13   L,R,x,y,mi:Integer;
14 begin{Compute};
15   for x:=1 to N do S[x]:=x;
16   L:=1; R:=N;
17   while L<R do begin
18     for x:=L+1 to R–1 do begin
19       mi:=Med3(S[L],S[x],S[R]);
20       if mi=S[L] then begin
21         y:=S[L]; S[L]:=S[x]; S[x]:=y;
22       end else if mi=S[R] then begin
23         y:=S[R]; S[R]:=S[x]; S[x]:=y;
24       end;
25     end{for x};
```

```
26        Inc(L); Dec(R);
27      end{while};
28
29      Compute:=S[L];
30  end{Compute};
```

A két szélső elem $n-2$ kérdéssel határozható meg, tehát a kérdések száma:

$$(n-2)+(n-4)+\ldots+3+1=\left(\frac{n-1}{2}\right)^2$$

Ha $n \leq 177$, akkor legfeljebb 7744 hívás kell, de ha $n \geq 179$, akkor legalább 7921. **Rendezést használó algoritmusok.**
Egy $\langle a_1, a_2, \ldots, a_m \rangle$ elemsorozatot rendezettnek nevezünk, ha

$$(\forall i,j,k)(1 \leq i < j < k \leq m)(Med3(a_i,a_j,a_k) = a_j)$$

Definiálhatnánk egy $x < y$ bináris lineáris rendezési relációt, amely lehetővé tenné, hogy bármely ismert rendezést használhassunk.

$$x,y,1,2$$
$$x,1,y,2$$
$$x,1,2,y$$
$$1,x,y,2$$
$$1,x,2,y$$
$$1,2,x,y$$

Ehhez azonban két Med3 hívás kell.
Rendezett sorozat előállítható ismételt beszúrással, indulva egy kételemű sorozattal.
A beszúrás helye meghatározható:
Lineáris kereséssel. A beszúrandó x elemet a sorozat két utolsó eleméhez hasonlítjuk
MED3(a_{m-1}, x, a_m) lekérdezéssel.
$\langle a_1, a_2, \ldots, a_{m-1}, a_m \rangle$
Bináris kereséssel. A beszúrandó x elemet a sorozat középső két eleméhez hasonlítjuk; MED3(a_{k-1}, x, a_k) lekérdezéssel.
$\langle a_{bal}, \ldots, a_{k-1}, a_k, \ldots, a_{jobb} \rangle$
Harmadoló kereséssel.
A beszúrandó x elemet a sorozat egy-harmad és kétharmad pozíciójában lévő két eleméhez hasonlítjuk; MED3(a_k, x, a_l) lekérdezéssel.
$\langle a_{bal}, \ldots, a_k, \ldots, a_l, \ldots, a_{jobb} \rangle$
1. Teljes rendezés algoritmus.
Képezzünk az $\{1, \ldots, n\}$ elemekből rendezett sorozatot:

$$S = \langle a_1, \ldots, a_n \rangle$$

A keresett medián a_m, $m = (n+1)/2$.
2. Felét rendező algoritmus.
Legyen $m = (n+1)/2$. Első lépésként állítsunk elő m elemből pl. az $\{1, \ldots, m\}$ elemekből rendezett sorozatot:

$$S = \langle a_1, \ldots, a_m \rangle$$

Minden további x elemet szúrjuk be az S sorozatba és hagyjuk el az utolsót. A keresett medián a sorozat utolsó eleme lesz.
3. Szűkítő rendezés algoritmus.
Legyen $m = (n+1)/2$. Első lépésként állítsunk elő m elemből pl. az $\{1, \ldots, m\}$ elemekből rendezett sorozatot:

$$S = \langle a_1, \ldots, a_m \rangle$$

Minden további x elemet szúrjuk be az S sorozatba

$$\overline{S} = \langle a_1, \ldots, a_i, x, a_{i+1}, \ldots, a_m \rangle$$

Nyilvánvaló, hogy $\overline{S}$ első eleme nem lehet a medián, mert van legalább m nálánál nagyobb elem. Hasonlóképpen $\overline{S}$ utolsó eleme sem lehet a medián, mert van legalább m nálánál kisebb.
Tehát hagyjuk el $\overline{S}$ első és utolsó elemét.

Beszúrási mód	Változat	Legrosszabb eset	Legjobb eset
Lineáris	Teljes	561749	282532
	Fél	421499	169655
	Szűkítő	281623	141676
Bináris	Teljes	12953	11680
	Fél	12477	11492
	Szűkítő	11481	10471
Harmadoló	Teljes	9399	8977
	Fél	9399	8522
	Szűkítő	8319	8041

1. táblázat. Med3 hívások száma $N = 1499$ elemre

```
Program Median4;
{Szűkítő rendezés }
Uses Device;
const
  MaxN=3000;                    {max. number of elements}
var
  N:Integer;                    {number of elements}
  M:Integer;                    {solution}
  S:array[1..MaxN] Of Integer;{sorted sequence}

Function FindPos(L,R,X:Word):Word; { Finds position of x by ternary search in the
ordered sequence S[L..R] } var
  L0,R0,Lm,Rm,mi,d,Xm:Word;
begin{FindPos}
  L0:=L; R0:=R;
  L:=L0-1; R:=R0+1;

  while (L+2<R) do begin
    d:=(R-L) Div 3;
    if (R-L) Mod 3=2 then  begin { partitions of length (d+1), d, (d+1) }
      Lm:=L+d+1; Rm:=R-(d+1);
    end else begin                { partitions of length d, (d+1), d }
      Lm:=L+d; Rm:=R-d;
    end;
    Xm:=Med3(S[Lm], S[Rm], X);
    if Xm=S[Lm] then
      R:=Lm
    else if Xm=S[Rm] then
      L:=Rm
    else begin
      L:=Lm; R:=Rm;
    end;
  end{while};

  if (R=L+2) then begin {handle extremal cases}
    if L=0 then L:=1;
    R:=L+1;
    Xm:=Med3(S[L], S[R], X);
    if Xm=S[L] then
      L:=L-1
    else if (Xm=S[R]) then
      L:=R
  end;

```

```
45    FindPos:=L;
46  end{FindPos};
47
48  Function Compute:Integer; var
49    L,R,x,xp,i,Mi:Integer;
50  begin{Compute};
51    Mi:=N div 2+1;
52    L:=1; R:=2;
53    S[1]:=1; S[2]:=2;
54    for x:=3 to Mi do begin   {build sorted sequence of elements 1..Mi}
55      xp:=FindPos(L,R,x);
56      for i:=R downto xp+1 do {insert x}
57        S[i+1]:=S[i];
58      S[xp+1]:=x;
59      Inc(R);
60    end{for};
61    {S[1..Mi] ordered }
62    for x:=Mi+1 to N do begin
63    {Invariant: There are N/2 elements greater then S[L] and
64                there are N/2 elements less then S[R].}
65      xp:=FindPos(L,R,x);
66      if (xp<L) then              {eliminate the two extemals:}
67        Dec(R)                    {one overflows on the left, eliminate the rightmost}
68      else if (xp=R) then
69        Inc(L)                    {one overflows on the right, eliminate the leftmost}
70      else begin                  {insert x into S[L..R]}
71        for i:=R downto xp+1 do {one overflows on the right}
72          S[i+1]:=S[i];
73        S[xp+1]:=x;
74        Inc(L);                   {eliminate the leftmost }
75      end;
76    end{for};
77
78    Compute:=S[R];                {L=R}
79  end{Compute};
80
81  begin{program}
82    N:=GetN;
83    M:=Compute;
84     Answer(M);
85  end.
```

4. Iterált harmadoló algoritmus.

Elvi algoritmus:

Legyen $H = \{1,2,\ldots,n\}$. Válasszunk két különböző tetszőleges $a,b \in H$ elemet, ezek lesznek a felosztó elemek. Majd H többi elemét osszuk három részbe:

$$H_1 = \{x : Med3(a,b,x) = a\} \tag{14}$$

$$H_2 = \{x : Med3(a,b,x) = x\} \tag{15}$$

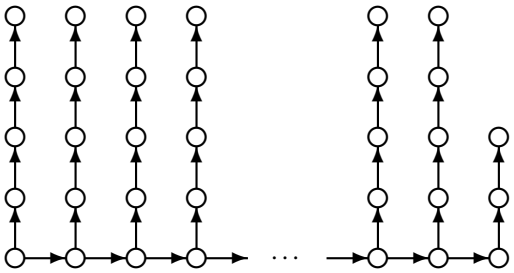
$$H_3 = \{x : Med3(a,b,x) = b\} \tag{16}$$

Tehát a keresést H_1, H_2, H_3 valamelyikében kell folytatni attól függően, hogy hány elem került az egyes halmazokba a felosztás során.

$$H_1 < a < H_2 < b < H_3$$

```
1  Function Keres(H:Halmaz; k:Integer):Integer;
2   {A H halmaz k-adik elemének keresése harmadoló felosztással }
3  Var H1,H2,H3:Halmaz;
```

```
4     a,b:Integer;
5     Procedure  Feloszt(H:Halmaz; Var H1,H2,H3:Halmaz; Var a,b:Integer);
6     Begin{Feloszt};
7       Valaszt(H,a); Valaszt(H,b);
8       Rendez(a,b); {a<b}
9       H1:=[]; H2:=[];H3:=[];
10      For x In H Do
11        Case Med3(a,x,b) of
12          x: H2:=H2+[x];
13          a: H1:=H1+[x];
14          b: H3:=H3+[x];
15        End{Case};
16    End{Feloszt};

1   Begin{Keres};
2     If |H|=1 Then Begin
3       Keres:=H eleme
4     End Else If |H|=2 Then Begin
5       Valaszt(H,a); Valaszt(H,b);
6       Rendez(a,b); {a<b}
7       If k=1 Then Keres:=a Else Keres:=b
8     End Else Begin
9       Feloszt(H,H1,H2,H3,a,b);
10      If k<=|H1| Then
11        Keres:=Keres(H1,k)
12      Else If k=|H1|+1 Then
13        Keres:=a
14      Else If (k>|H1|+1) And (k<=|H1|+|H2|+1) Then
15        Keres:=Keres(H2,k-|H1|-1)
16      Else If k=|H1|+|H2|+1 Then
17        Keres:=b
18      Else
19        Keres:=Keres(H3,k-(|H1|+|H2|+2)
20    End;
21  End{Keres};

22  Program Median5;
23  {Iterált harmadoló felosztás algoritmus}
24  uses Device;
25  const
26    MaxN=3000;
27  var
28    N:Integer;
29    M:Integer;{a megoldás}
30    S:array[1..MaxN] Of 0..MaxN;
31
32  Function Keres(K:Integer):Integer; var Veg1,Veg2,Veg3:Integer;
33    Bal,Jobb,a,b,i:Integer;
34    H1,H2,H3:Integer;
35    m12a,m1ab:Integer;
36
37    procedure  Feloszt(Bal,Jobb:Integer; var V1,V2,V3,a,b:Integer);
38    var i,X,m:Integer;
39    begin{Feloszt};
40      a:=S[Bal]; b:=S[Bal+1];
41      if Bal>1 then begin{a~b rendezése 1->2 höz képest}
42        m12a:=Med3(1,2,a);
43        m1ab:=Med3(1,a,b);
```



7. ábra. 2-dimenziós rendezett lista