Számjáték

Tekintsük az alábbi alábbi kétszemélyes játékot. A játék úgy kezdődik, hogy véletlenszerűen leraknak az asztalra n darab pozitív egész számot egy sorba. A két játékos felváltva lép. Egy lépés azt jelenti, hogy a játékos leveszi az asztalról a számsorozat bal avagy jobb végén lévő számot. A kiválasztott számot törlik a tábláról és a játékos pontszámához adódik. A játék akkor ér véget, ha a számok elfogytak.
Mindkét játékos arra törekszik, hogy a lehető legtöbb pontot gyűjtse. A játékot az első játékos kezdi.

Feladat

Írjunk olyan programot, amely az kezdő játékos szerepét játssza! A második játékos lépéseit egy már adott számítógépes program szolgáltatja. A két játékos a rendelkezésre bocsátott `Play` modul műveleteinek felhasználásával kommunikál egymással.

Könytári műveletek

TablaMeret a kezdeti játékállásban a táblán lévő számok számát adja ami páros szám. Minden más Play modul műveletet meg kell előznie egy `TablaMeret` végrehajtásának. A táblaméret legfeljebb 1000 lehet.

Tabla(i) a kezdeti játékállásban a táblán lévő i-edik számot adja.

Lepesem(L) hívással közli az első játékos a választását; ha balról vesz le, akkor az `L` aktuális paraméter értéke a `B` karakter, ha jobbról, akkor a `J` karakter legyen.

Lepesed a második játékos (a gép) utolsó lépését adja, a `B` karaktert, ha balról választott számot, a `J` karaktert, ha jobbról.

A Play modul műveletei Pascal nyelv esetén

- `function TablaMeret:integer;`
- `function TablaMeret:integer;`
- `procedure Lepesem(L:char);`
- `function Lepesed:char;`

A Play modul műveletei C/C++ nyelv esetén

- `int TablaMeret();`
- `int Tabla(int i);`
- `void Lepesem(char L);`
- `char Lepesed();`

Feltételek

- n páros szám és $2 \leq n \leq 1000$.
- Programod nem írhat és nem olvashat egyetlen fájlt sem, beleértve a standard bemenetet és kimenetet!

Gyakrolás

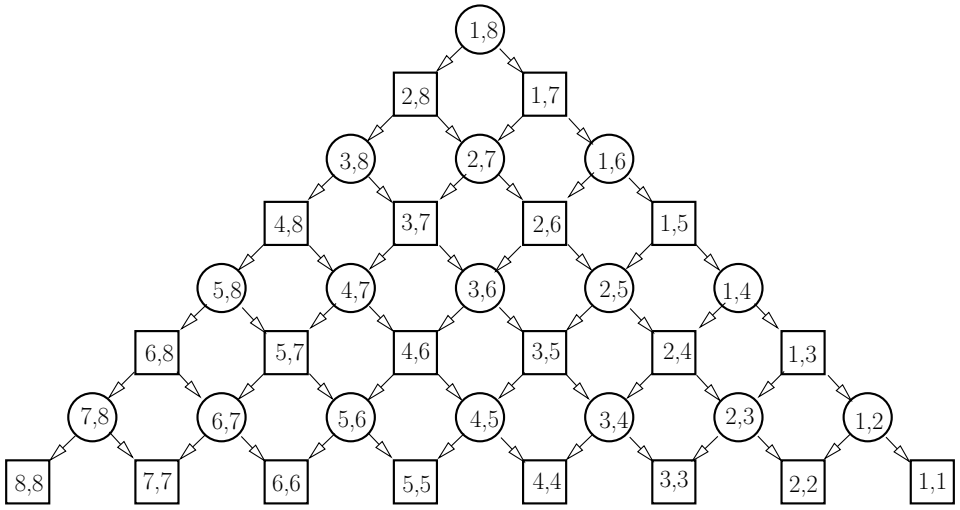
A minta.zip letölthető állomány tartalmazza a Play modulnak egy megvalósítását Pascal és C++ nyelven.
Pascal esetén elegendő csak a `uses Play;` import direktíva megadása a program elején.
C/C++ esetén projektet célszerű készíteni és a Play.h és Play.cpp állományokat is hozzá kell adni a projekthez.
A program végrahajtásakor a standard bemenetről olvassa először az n értékét, majd a kezdeti játékállást megadó számokat. A program hibátlan kommunikáció esetén a standard kimenetre két számot ír ki, az első az első játékos összpontszáma, a második pedig a második játékosé. A letölthető minta Play modulban a második játékos nem optimálisan játszik, hanem mohó módon mindig a nagyobbikat választja!

Példa

Bemenet	Kimenet
6 4 7 2 9 5 2	18 6

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.

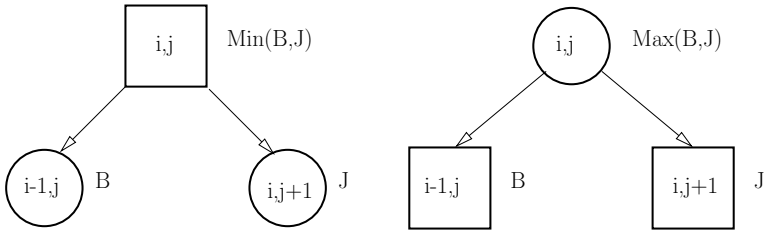


1. ábra. A játékállások gráfja $n = 8$ esetén. Körrel jelölt állásból ($i + j$ páratlan) az első, négyzettel jelölt állásból ($i + j$ páros) a második játékos lép.

Definiáljuk minden (i, j) játékállásra azt a maximális pontszámot, amit az első játékos elérhet ebből a játékállásból indulva, feltéve, hogy a második játékos is arra törekszik, hogy a lehető legtöbb pontot szerezze. Jelölje ezt az értéket $Opt(i, j)$. $Opt(i, j)$ a következő rekurzív összefüggés számítható.

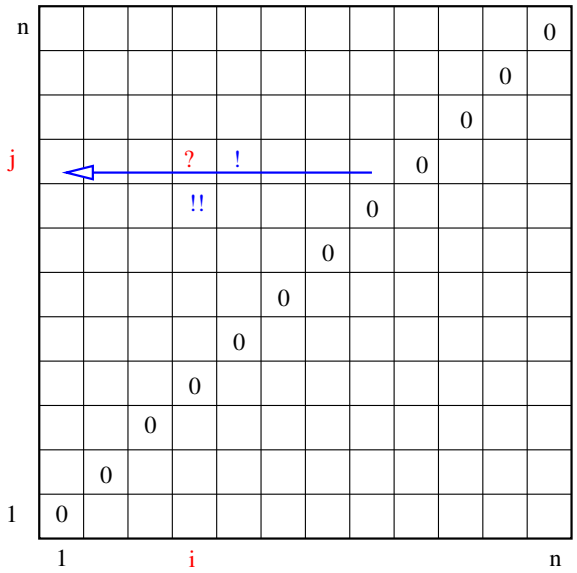
$$Opt(i,j) = \begin{cases} 0 & \text{ha } i = j \\ \max(a_i + Opt(i+1,j), a_j + Opt(i,j-1)) & \text{ha } i < j \text{ és } i+j \text{ páratlan} \\ \min(Opt(i+1,j), Opt(i,j-1)) & \text{ha } i < j \text{ és } i+j \text{ páros} \end{cases}$$

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



2. ábra. Mini-max szabály.

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



3. ábra. Táblázatkitöltési sorrend. $Opt(i, j)$ összetevői: $Opt(i+1, j)$ és $Opt(i, j-1)$

Megvalósítás C++ nyelven

```
#define maxN 1001
#include "Play.h"    //a 2. játékost megvalósító modul műveletei
int N;               //táblaméret
int A[maxN];         //a táblán lévő számok sorozata
char Lep[maxN][maxN];//az 1. játékos optimális lépései
int Opt[maxN][maxN]; //az 1. játékos Opt[i][j] pontot szerezhet (i,j) játékállásból

void Elofeldolgoz(){
   int PontBal,PontJobb;
   for(int j=1; j<=N;j++){
      Opt[j][j]=0;
      for(int i=j-1; i>0; i--)
      if((i+j)%2==1){//1. játékos lép
         PontBal=A[i]+Opt[i+1][j];
         PontJobb=A[j]+Opt[i][j-1];
         if(PontBal>PontJobb){
            Opt[i][j]=PontBal;
            Lep[i][j]='B';
         }else{
            Opt[i][j]=PontJobb;
            Lep[i][j]='J';
         }
      }else{//2. játékos lép
         if(Opt[i+1][j]<Opt[i][j-1])
            Opt[i][j]=Opt[i+1][j];
         else
            Opt[i][j]=Opt[i][j-1];
      }
   }
}
void Jatszas(){
   int Bal=1,Jobb=N;
   char L2;
   while(Bal<Jobb){
      if(Lep[Bal][Jobb]=='B'){
         Bal++;
         Lepesem('B');
      }else{
         Jobb--;
         Lepesem('J');
      }
      L2=Lepesed();//a válaszlépés lekérdezése
      if(L2=='B')
         Bal++;
      else
         Jobb--;
   }
}
int main(){
   N=TablaMeret();
   for (int i=1;i<=N;i++) A[i]=Tabla(i);
   Elofeldolgoz();
   Jatszas();
}
```