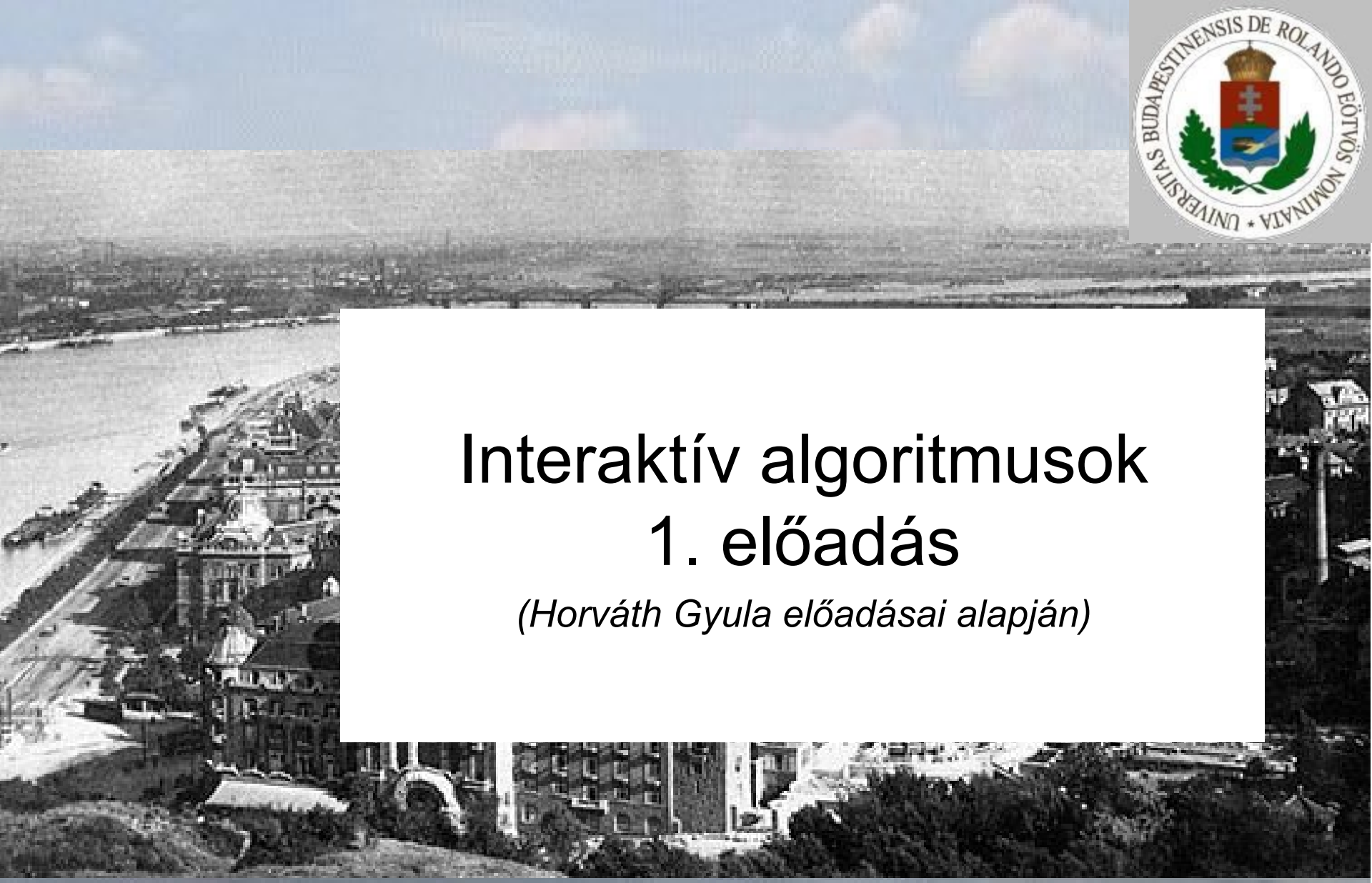


Interaktív algoritmusok
1. előadás

(Horváth Gyula előadásai alapján)

Interaktív algoritmusok

Az interaktív algoritmusok lényege:

- a program párbeszédet folytat egy másik programmal (eljárásokkal) a feladat megoldása érdekében;
- nem a teljes bemenet beolvasása után kell eredményt közölnie.

Megvalósítási lehetőségei:

- főprogramot írunk, amely adott eljárásokat hív meg;
- eljárásokat írunk (egymással közös változókon keresztül kommunikálhatnak), amelyeket adott főprogram hív meg.

Mindkettő közös jellemzője: az általunk megírandó programrészben sem beolvasás, sem kiírás nincs.

Interaktív algoritmusok

Az interaktív algoritmusok fajtái:

- online algoritmusok;
- kitalálós feladatok;
- kétszemélyes játékok.

Online algoritmusok

Jellemzői:

- a bemenetet csak részenként ismerjük,
- minden lépésben az addig megkapott információ alapján kell döntenünk.

Korábban ismert online (vagy azzá tehető) algoritmusok:

- beillesztéses rendezés
- online feszítőfa
- online Floyd-Warshall

Online algoritmusok – benzinkút

Feladat:

Egy benzinkútnál K helyen lehet tankolni. Minden tankolás pontosan T percig tart. Autók érkeznek adott időpontokban a benzinkúthoz. Minden autós a leghamarabb szeretne tankolni.

Adjuk meg minden autóra a tankolása befejezése időpontját!

A megoldás elve: Tároljuk minden helyhez az utolsó tankoló autó távozási idejét! Az új autó abba a sorba áll, amelyiknél ez az idő a lehető legkisebb.

Első változat: eljárásokat kell meghívni egy főprogramból.

Online algoritmusok – benzinkút

`KutSzam`, egyszer kell hívni a program elején, a tankolási helyek számát adja.

`Ido`, egyszer kell hívni az előző függvény után, az autók tankolási idejét adja.

`Erkezik()`, az érkezési sorrendben következő autó benzinkúthoz érkezési idejét adja.

`Tavozik(x)`, minden érkezés után meg kell hívni, paramétere az autó leghamarabbi távozási ideje legyen! Az utolsó érkező autóra meghívása után nem tér vissza a hívó eljárásba.

Online algoritmusok – benzinkút

```
k:=KutSzam; t:=Ido; u:=(0,...,0)
Ciklus
  x:=Erkezik
  min:=1
  Ciklus i=2-től k-ig
    Ha u(i)<u(min) akkor min:=i
  Ciklus vége
  Ha u(min)<x akkor u(min):=x+t
            különben u(min):=u(min)+t
  Tavozik(u(min))
Ciklus vége
```

Online algoritmusok – benzinkút

Második változat: eljárásokat kell írni, amiket majd egy főprogram fog meghívni.

`KutSzam(N)`, először ezt hívják meg, a tankolási helyek számát kapja paraméterként.

`Ido(T)`, másodszor ezt hívják meg, az autók tankolási idejét kapja paraméterként.

`Erkezik(x)`, paraméterében az érkezési sorrendben következő autó benzinkúthoz érkezési idejét kapja, a függvény értéke az autó leghamarabbi távozási ideje legyen!

Online algoritmusok – benzinkút

```
Változó k,t: Egész
        u: tömb(1..maxk,Egész)
Kutszam(N):
  k:=N; u:=(0,...,0)
Eljárás vége.
Ido(i):
  t:=i
Eljárás vége.
```

Online algoritmusok – benzinkút

```
Erkezik(x):
  min:=1
  Ciklus i=2-től k-ig
    Ha u(i)<u(min) akkor min:=i
  Ciklus vége
  Ha u(min)<x akkor u(min):=x+t
            különben u(min):=u(min)+t
  Erkezik:=u(min)
Függvény vége.
```

Online algoritmusok – összeadás

Feladat:

Két sokjegyű természetes számot számjegyenként balról jobbra is össze lehet adni. Ez azt jelenti, hogy az eredmény számjegyei is balról jobbra állnak elő, mindig abban a pillanatban, amikor már tudjuk, hogy a tőle jobbra levők miatt nem változhatnak meg.

Megoldási elv: Tegyük fel, hogy az első szám a hosszabb, a másodikat egészítsük ki gondolatban elölről 0-kkal!

Példák:

```
  123     355     35552     66654     567
+235    +245    +24443    +33445    +756
 358     600     59995    100099    1323
```

Online algoritmusok – összeadás

Ha az adott pozíció utáni két számjegy összege kisebb 9-nél, akkor átvitellel sem lehet 9-nél nagyobb, azaz az adott pozícióra ismerjük az eredmény értékét, kiírhatjuk.
Ha az összegük pontosan 9, akkor elképzelhető, hogy az adott pozícióra lesz átvitel, azaz őrizzük meg, hogy hány helyen volt az összeg 9 – számoljuk a 9-esek számát és várunk a kiírással.
Ha pedig nagyobb 9-nél, akkor lesz átvitel, az összes korábbi 9-esen keresztül terjeszteni kell az átvitelt – azaz a 9-esek helyére 0-kat írunk, az előttük levő számjegyet pedig eggyel megnöveljük.

Online algoritmusok – összeadás

`getaN`: a program elején egyszer kell meghívni, megadja az első szám számjegyei számát;
`getbN`: a program elején egyszer kell meghívni, megadja a második szám számjegyei számát;
`geta`: megadja az első szám következő számjegyének értékét – balról jobbra haladva;
`getb`: megadja a második szám következő számjegyének értékét – balról jobbra haladva;
`kiir(c)`: az eredmény következő számjegyét írja – balról jobbra haladva. Az utolsó kiírás után -1-gyel kell meghívni!

Online algoritmusok – összeadás

```
n:=getaN; m:=getbN
c:=0; db:=0; volt:=hamis
Ciklus i=n-től 1-ig -1-esével
  a:=geta
  Ha i≤m akkor b:=getb különben b:=0
  Ha a+b<9 akkor {nincs átvitel}
    Ha volt akkor kiir(c)
    Ciklus db-szer: kiir(9)
    c:=a+b; db:=0; volt:=igaz
  különben ha a+b=9 akkor {lehet innen átvitel}
    Ha i=n akkor c:=0
    db:=db+1
  különben ...{a+b>9, biztos van átvitel}
```

Online algoritmusok – összeadás

```
 különben ...{a+b>9.biztps van átvitel}
    kiir(c+1)
    Ciklus db-szer: kiir(0)
    c:=a+b-10; db:=0; volt:=igaz
Ciklus vége
Ha volt vagy c>0 akkor kiir(c)
Ciklus db-szer: kiir(9)
kiir(-1)
```

Kitalálós játékok

A következő típusú kitalálós játékokat nézzük:

- a kitalálandó értékek 1 és N közötti egész számok;
- az egyes kérdésekhez költségeket rendelhetünk – ha ez konstans, akkor minimális számú kérdést feltéve kell kitalálnunk a számot; ha nem konstans; akkor pedig minimális költséggel;
- blöffölni nem lehet, csak akkor lehet megadni a kitalálandó értéket, ha az egyértelmű;
- a válaszadó „csalfa”, azaz mindig úgy válaszol, hogy a lehető legtovább tartson a választ megtalálni.

Kitalálós játékok

Feladat:

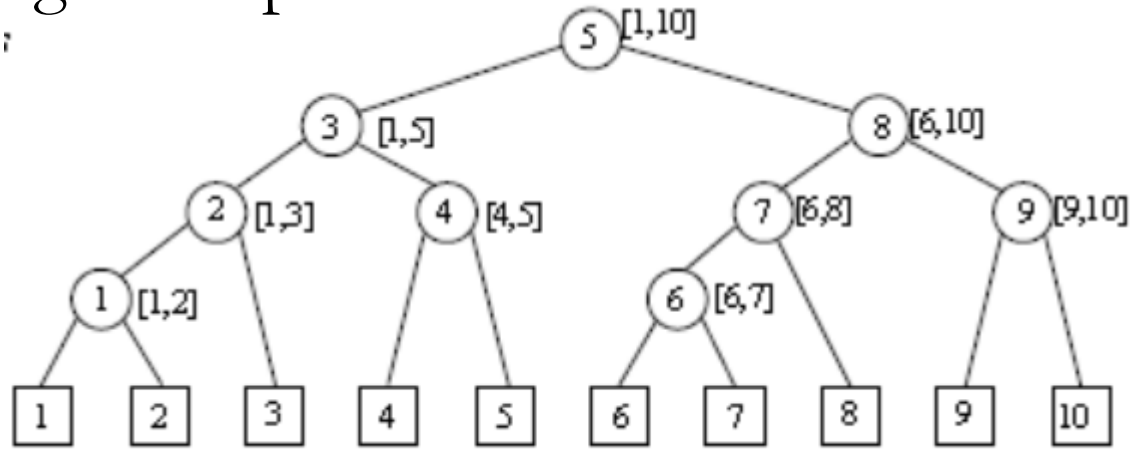
A válaszadó kitalál egy 1 és N közötti számot. A lehetséges kérdéseink: A kitalálandó szám kisebb vagy egyenlő-e, mint X? A számot a lehető legkevesebb kérdéssel kell kitalálni!

Megoldás: alkalmazzuk a logaritmikus keresés elvét, a lehetséges számintervallumból mindig a középsőt kérdezzük!

Kitalálós játékok

Felépítünk egy kérdezőfát. A kérdezőfa leveleiben a kitalálandó értékek szerepelnek, a nem levél elemekben pedig a kérdésként felteendő kérdések. A legkevesebb kérdésből kitalálás azt jelenti, hogy a fa magassága a lehető legkisebb legyen!

Ebben az esetben a kérdezőfát csak elképzelni kell, felépíteni nem szükséges, mivel minden intervallumra a felteendő kérdésben a két határ átlaga szerepel.



Kitalálós játékok

GetN : Pontosan egyszer kell hívni a program elején és a visszaadott érték az Éva által gondolt szám maximuma.

Kérdés(x): Igaz értékű, ha a gondolt szám kisebb vagy egyenlő, mint x.

Megoldás(x): A kitalált számot ezzel a művelettel kell közölni.

Online algoritmusok – összeadás

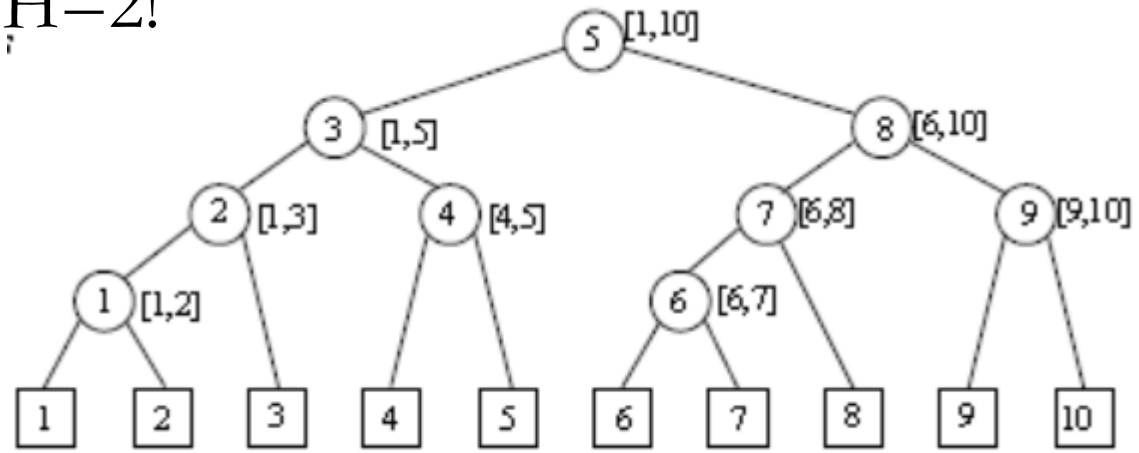
```
n:=GetN; e:=1; u:=n; k:=(e+u)/2
Ciklus amíg e<u
   Ha Kérdés(k) akkor u:=k különben e:=k+1
   k:=(e+u)/2
Ciklus vége
Megoldás(k)
```

Kitalálós játékok

Feladat:

A válaszadó kitalál egy 1 és N közötti számot. A lehetséges kérdéseink: A kitalálandó szám kisebb vagy egyenlő-e, mint X? A számot a lehető legkevesebb kérdéssel kell kitalálni, de maximum H-szor szabad NEM választ kapnunk!

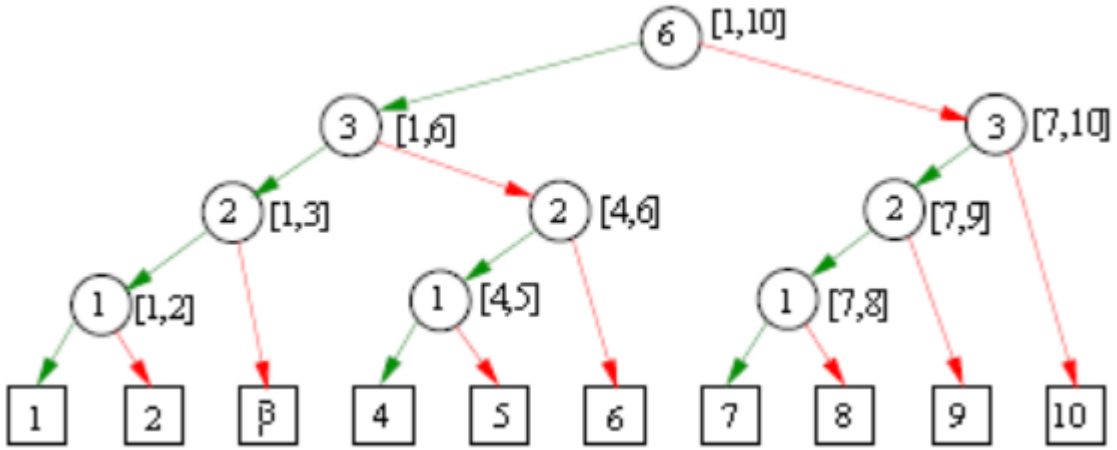
Megoldás: itt is a részekre osztás a jó, de az előző megoldás az alábbi esetre nem jó, ha H=2!



Kitalálós játékok

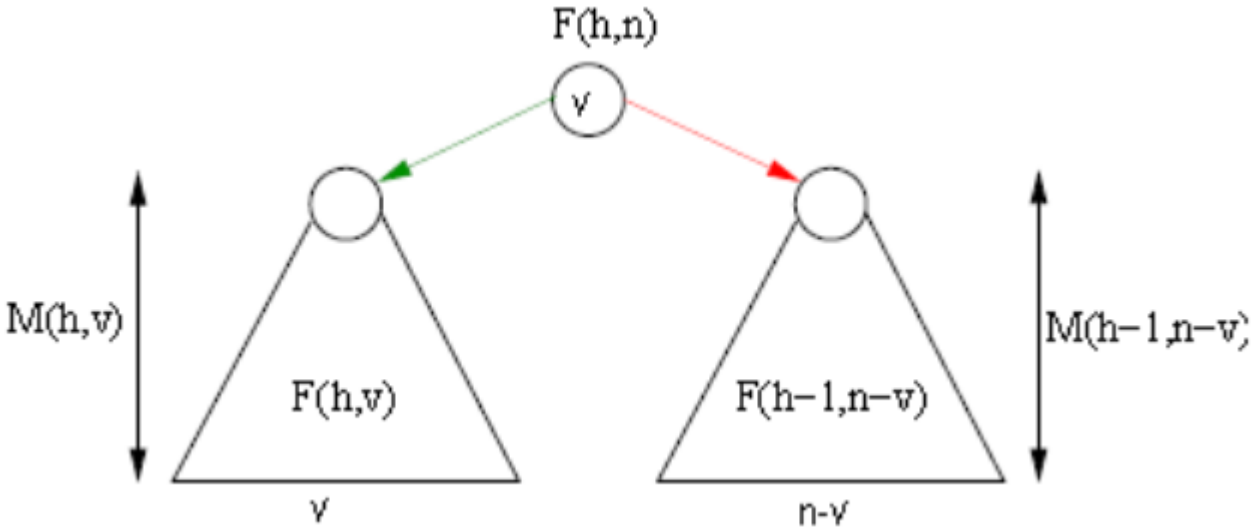
Felépítünk egy kérdezőfát. A kérdezőfa leveleiben a kitalálandó értékek szerepelnek, a nem levél elemekben pedig a kérdésként felteendő kérdések. A legkevesebb kérdésből kitalálás azt jelenti, hogy a fa magassága a lehető legkisebb legyen!

A legfeljebb H-szor kaphatunk NEM választ pedig azt jelenti, hogy a fában bármely a gyökértől valamelyik levélig vezető úton maximum H-szor léphetünk jobbra.



Kitalálós játékok

A kérdezőfa felépítése: dinamikus programozás – számítsuk ki az N elemre H hibázó optimális kérdezőfa magasságát (M(h,n)), illetve állítsuk elő magát a fát! F(h,n) a felteendő kérdés.



$$M(h,n)=\begin{cases} 0 & ha\ \ n=1 \\ \min\limits_{v=1}^{n-1}\left(\max\left(M(h,v),M(h-1,n-v)\right)+1\right) & ha\ \ n>1 \end{cases}$$

Kitalálós játékok

GetN : Pontosan egyszer kell hívni a program elején és a visszaadott érték a gondolt szám maximuma.

GetH : Az előző után, egyszer kell hívni, a visszaadott érték a NEM válaszok maximális száma.

Kérdés(x): Igaz értékű, ha a gondolt szám kisebb vagy egyenlő, mint x.

Megoldás(x): A kitalált számot ezzel a művelettel kell közölni.

Online algoritmusok – összeadás

```
Előkészítés:
 M(1..h,1):=(0,...,0); F(1..h,1):=M(1..h,1)
 M(1,1..n):=(0,1,...,n-1); F(1,1..n):=M(1,1..n)
 Ciklus j=2-től h-ig
  Ciklus i=2-től n-ig
   Ha i=j akkor M(j,i):=M(j-1,i); F(j,i):=F(j-1,i);
   különben
     ...
```

$$M(h,n)=\begin{cases} 0 & ha\ \ n=1 \\ \min\limits_{v=1}^{n-1}\left(\max\left(M(h,v),M(h-1,n-v)\right)+1\right) & ha\ \ n>1 \end{cases}$$

Online algoritmusok – összeadás

```
  ...
  különben
    min:=+∞; minv:=0
    Ciklus v=1-től i-1-ig
     Ha M(j,v)>M(j-1,i-v) akkor Mjiv:=M(j,v)
                          különben Mjiv:=M(j-1,i-v)
     Ha Mjiv<min akkor min:=Mjiv; minv:=v
    Ciklus vége
    M(j,i):=min+1; F(j,i):=minv
  Ciklus vége
Ciklus vége
Eljárás vége
```

$$M(h,n)=\begin{cases} 0 & ha\ \ n=1 \\ \min\limits_{v=1}^{n-1}\left(\max\left(M(h,v),M(h-1,n-v)\right)+1\right) & ha\ \ n>1 \end{cases}$$

Online algoritmusok – összeadás

```
n:=GetN; h:=GetH; Előkészítés
e:=1; u:=n; k:=e+F(h,u-e+1)-1   {k:=F(h,n)}
Ciklus amíg e≤n
  Ha Kérdés(k) akkor u:=k
               különben e:=k+1; h:=h-1
  k:=e+F(h,u-e+1)-1
Ciklus vége
Megoldás(k)
```

A kérdezőfa mindig bináris fa? Nem! Ha pl. olyan kérdést tehetünk fel, hogy a kitalálandó érték az [a,b] intervallumban van-e, arra háromféle választ kaphatunk: IGEN, <a, >b.

Kitalálós játékok

Egy különleges kitalálós játékban előre leírt kérdésekkel, a válaszok előzetes ismerete nélkül kell kitalálni a megoldást.

Itt előre fel kell tenni az összes kérdést. A megoldás itt az lesz, hogy az i. kérdésben megkérdezzük, hogy a gondolt szám i. bitje 1-e? Mivel egy N számnak $\lceil \log(N) \rceil$ bitje van, így az adott szám log(N) lépésben kitalálható.

Sajnos ez a stratégia azonban nem alkalmazható a kitalálós játékok széles körére.

Kétszemélyes játékok

Többféle kétszemélyes játék lehet. Először a következő típusú kétszemélyes játékokat nézzük:

- a játékosok felváltva lépnek;
- egyértelmű szabály határozza meg a lehetséges lépéseket;
- a játékállások száma véges;
- a játék véges lépés megtételével befejeződik (játékállás nem ismétlődhet);
- minden játékállásból mindkét játékosnak ugyanazok a lehetőségei;
- **az a játékos veszít, aki nem tud lépni;**
- mindkét játékos arra törekszik, hogy nyerjen, azaz ha tud, akkor úgy lép, hogy nyerjen.

Kétszemélyes játékok

Nyerő állás: amiből a kezdő játékos biztosan nyer,

Vesztő állás: amiből a kezdő játékos nem tud nyerni.

Stratégia: A kezdő játékosnak úgy kell lépnie, hogy a lépése vesztő állást hozzon létre, azaz ilyenkor az ellenfél bármit is lép, újra nyerő állás alakul ki.

Kétszemélyes játékok

Képzeljük el a játékot egy gráfként! A gráf pontjai legyenek a lehetséges játékállások, élei pedig az egyes játékállásokban alkalmazható lépések, amelyek más játékállásokba vezetnek.

Mivel játékállás nem ismétlődhet, ez a gráf egy körmentes irányított gráf lesz.

Színezzük ki ezt a gráfot két színnel, piros legyen a vesztő, zöld pedig a nyerő állások színe!

A 0 kifokú pontok a vesztő állások, a többit ezekből kiindulva számíthatjuk.

Megjegyzés: a gráfot magát általában nem állítjuk elő, hanem az algoritmus közben számítjuk. Sajnos így is nagyon nagy lehet!

Kétszemélyes játékok

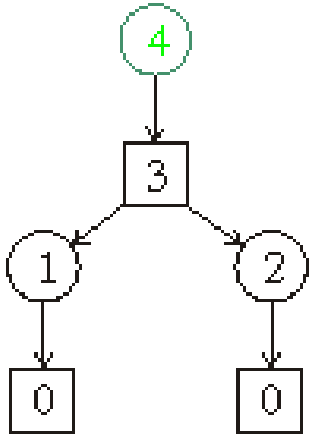
Fibonacci NIM játék

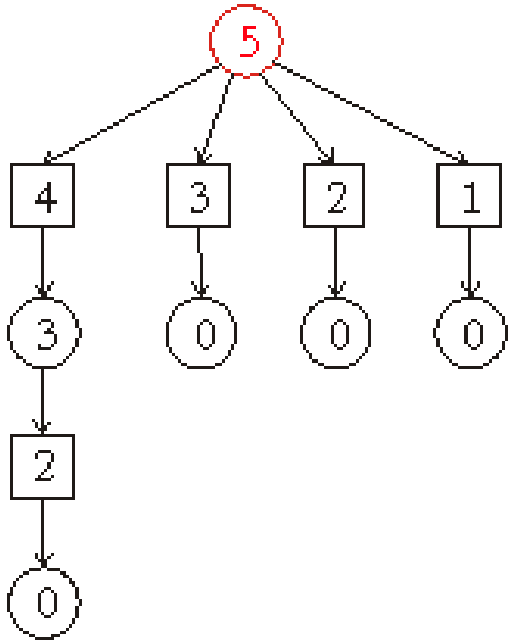
Adott egy kavics halom, amelyben N kavics van Két játékos felváltva vehet el valahány kavicsot. A kezdő játékos tetszőleges számú kavicsot elvehet, de az összest nem. Legalább 1 kavicsot minden lépésben el kell venni. Minden további lépésben mindegyik játékos legfeljebb az előző lépésben elvett kavicsok számának a kétszeresét veheti el. A játékot az nyeri, aki az utolsó kavicsot veszi el.

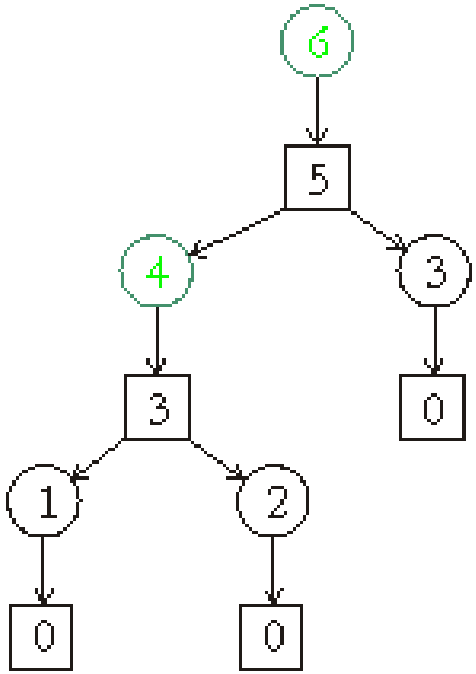
A lehetséges N-ek elemzésével kitalálható, hogy a vesztő állásokban N Fibonacci szám.

Kétszemélyes játékok

N=2 és 3 esetén a vesztő állás nyilvánvaló, nézzük meg a játékfát N=4,5,6 esetére:







Kétszemélyes játékok

A Zeckendorf tétel alapján minden természetes szám felbontható nem szomszédos Fibonacci számok összegére.

Ha az N felbontásában szereplő legkisebb Fibonacci számnyi kavicsot veszünk el először, akkor az ellenfél biztosan nem tud Fibonacci szám pozícióba lépni (mert két nem szomszédos Fibonacci szám közül a második biztosan nagyobb, mint az első kétszerese).

Kétszemélyes játékok

GetN: Pontosan egyszer kell hívni a program elején, értéke a kavicsok száma kezdetben.

Elvesz(x): Az ellenfél válasza arra, ha x kavicsot elveszel a halomból. A kapott egész szám értéke legalább 1 és legfeljebb 2*x. Az x értéke (az első hívás kivételével) nem lehet több mint az ellenfél által előzőleg elvett kavicsok számának két-szerese. Ha a te lépéseddel vagy az ellenfél lehetséges lépésével a játék véget ér, a programod ebben a függvényben automatikusan leáll.

Kétszemélyes játékok

```
n:=GetN; Fibonacci_előállítás; k2:=(n-1)div 2
Ciklus amíg n>0
   k:=Zeckendorf(n)
   Ha 2*k2<k akkor k:=1 {nem tudok annyit elvenni}
   n:=n-k
   k2:=elvesz(k); n:=n-k2
   Ha n≤2*k2 akkor k2:=elvesz(n) {a maradék mind elvehető}
Ciklus vége
```

Kétszemélyes játékok

```
Zeckendorf(x):
  i:=0; z:=x
  Ciklus amíg Fib(i)<x
    i:=i+1
  Ciklus vége
  Ciklus amíg x>0
    Ciklus amíg x<Fib(i)
      i:=i-1
    Ciklus vége
    x:=x-Fib(i); z:=Fib(i); i:=i-2
  Ciklus vége
  Zeckendorf:=z
Függvény vége.
```

Az x Fibonacci felbontásában szereplő legkisebb Fibonacci számot adja.

Kétszemélyes játékok

Most a következő típusú kétszemélyes játékokat nézzük:

- a játékosok felváltva lépnek;
- egyértelmű szabály határozza meg a lehetséges lépéseket;
- a játékállások száma véges;
- a játék véges lépés megtételével befejeződik (játékállás nem ismétlődhet);
- minden játékállásból mindkét játékosnak ugyanazok a lehetőségei;
- **A cél a minél nagyobb pontszám elérése;**
- mindkét játékos arra törekszik, hogy nyerjen, azaz ha tud, akkor úgy lép, hogy nyerjen.

Kétszemélyes játékok

Feladat:

A játéktábla pozitív egész számok páros hosszú sorozata. A két játékos felváltva a 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 játékot az első játékos kezdi.

Ebben az esetben a kezdő játékosnak van nyerő stratégiája → a stratégia alapján kell a programot megírni!

Kétszemélyes játékok

Jelölje $(a_1,...,a_n)$ a kezdeti játékállást! Minden lehetséges játékállást egyértelműen meghatározza az, hogy mely számok vannak még a táblán. Minden játékállás azonosítható (i; j) számpárral, ami azt jelenti, hogy a táblán az $(a_i,...,a_j)$ számsorozat van. Mivel n páros szám, így minden esetben, amikor az első játékos lép, vagy i páros és j páratlan, vagy fordítva, azaz 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.

Kétszemélyes játékok

`StartGame(A,n)`: Az első játékos a játszmát ennek végrehajtásával indítja.

`MyMove(C)`: Ha az első játékos a bal oldalról választ számot, akkor a C=’L’, ha jobbról, akkor a C=’R’ paraméterrel hívja.

`YourMove`: Az első játékos a gép lépését ezzel tudhatja meg. A függvény értéke ’L’ vagy ’R’ lesz attól függően, hogy a második játékos a bal vagy a jobb oldalról választott.

Kétszemélyes játékok

```
Játék:
  StarGame(A,n); PS:=0; PT:=0
  Ciklus i=1-től n-ig
    Ha páros(i) akkor PS:=PS+A(i)
                különben PT:=PT+A(i)
  Ciklus vége
  P:=PS>=PT
  i:=1; j:=n
  ...
```

Kétszemélyes játékok

```
  ...
  Ciklus amíg i<=j
    Ha P akkor Ha páros(i)
                  akkor MyMove('L'); i:=i+1
                 különben MyMove('R'); j:=j-1
    különben Ha páros(i)
                  akkor MyMove('R'); j:=j-1
                 különben MyMove('L'); i:=i+1
    C:=YourMove
    Ha C='L' akkor i:=i+1 különben j:=j-1
  Ciklus vége
Program vége.
```

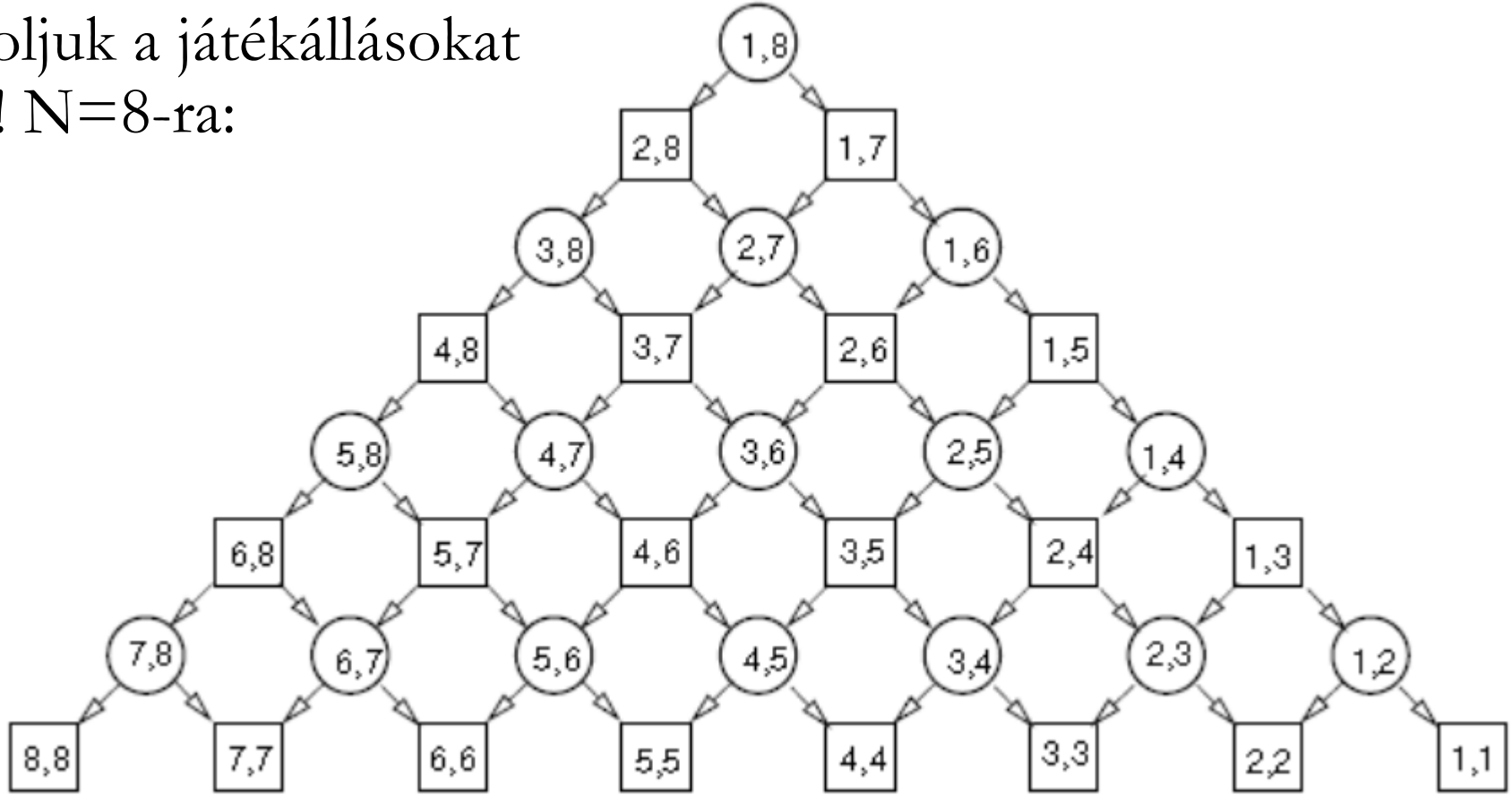
Kétszemélyes játékok

Feladat:

A játéktábla pozitív egész számok sorozata. A két játékos felváltva a 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 játékot az első játékos kezdi. A cél, hogy az első játékos a lehető legtöbbet szerezze meg, feltéve, hogy erre törekszik a második játékos is.

Kétszemélyes játékok

Ábrázoljuk a játékállásokat gráffal! N=8-ra:



Körrel jelölt állásból (i+j páratlan) az első, négyzettel jelölt állásból (i+j páros) a második játékos lép.

Kétszemélyes játékok

Definiáljuk minden (i,j) játékállásra azt a maximális pontszámot, amit az első játékos elérhet ebből a játékállásból indulva (dinamikus programozás)! **Minimax elv**: én maximálisat szeretnék nyerni, az ellenfelem pedig ezt minimalizálni szeretné!

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

Legyen Lép(i,j)= 'L', ha $a_i+Opt(i+1,j)>a_j+Opt(i,j-1)$, egyébként pedig 'R'!

Kétszemélyes játékok

```
Előkészítés:
 Ciklus i=n-től 1-ig -1-esével
  Opt(i,i):=0
  Ciklus j=i+1-től n-ig
   Ha páros(i+j) akkor {2. játékos}
      Ha Opt(i+1,j)<Opt(i,j-1)
         akkor Opt(i,j):=Opt(i+1,j)
      különben Opt(i,j):=Opt(i,j-1)
   különben {1. játékos}
   ...
```

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

Kétszemélyes játékok

```
   ...
   Bal:=a(i)+Opt(i+1,j); Jobb:=a(j)+Opt(i,j-1)
   Ha Bal>Jobb akkor Opt(i,j):=Bal; Lép(i,j):='L'
      különben Opt(i,j):=Jobb; Lép(i,j):='R'
  Ciklus vége
 Ciklus vége
Eljárás vége.
```

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

Kétszemélyes játékok

```
Játék:
  StartGame(A,n); Előkészítés
  i:=1; j:=n
  Ciklus amíg i≤j
    MyMove(Lép(i,j))
    Ha Lép(i,j)='L' akkor i:=i+1 különben j:=j-1
    C:=YourMove
    Ha C='L' akkor i:=i+1 különben j:=j-1
Eljárás vége.
```

Interaktív algoritmusok
1. előadás vége