Programozás
11. előadás

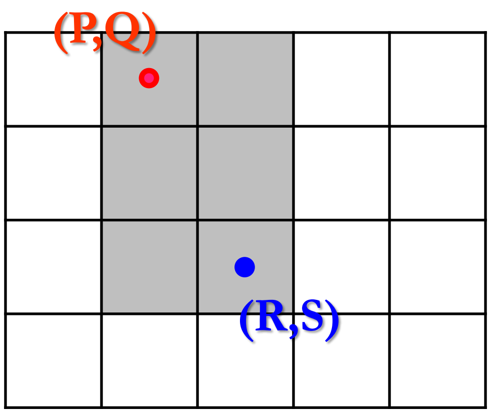
Tartalom

- Hatékony algoritmikus technikák
  - Segédösszegek számítása
  - Ablakozás (kétféleképpen)
  - Változásfigyelés
  - Intervallum-manipulációk (háromféleképpen)
- Rekurzió
  - Rekurzió és iteráció
  - Programozási tételek rekurzívan

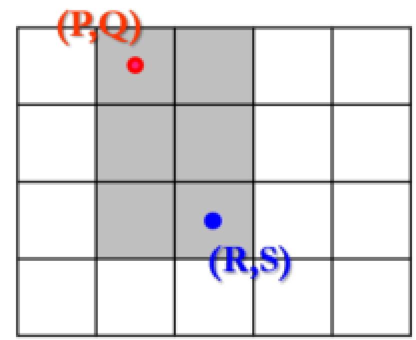
Segédösszegek

Egy földműves egy téglalap alakú területet szeretne vásárolni egy **N×M**-es téglalap alakú földterületen. Tudja minden megvásárolható földdarabról, hogy azt megművelve mennyi lenne a haszna vagy a vesztesége.

Adjuk meg azt a téglalapot, amelyen a legnagyobb haszon érhető el!



Segédösszegek



- Bemenet: $N,M \in \mathbb{N}$, $T_{1..N,1..M} \in \mathbb{Z}^{N \times M}$
- Kimenet: $P,Q,R,S \in \mathbb{N}$
- Előfeltétel: –
- Utófeltétel: $1 \le P \le R \le N$ és $1 \le Q \le S \le M$ és
  $\forall i,j,k,l\ (1 \le i \le k \le N, 1 \le j \le l \le M)$: érték$(P,Q,R,S) \ge$ érték(i,j,k,l)
- Definíció: érték: $\mathbb{N}^4 \to \mathbb{Z}$

$$\text{érték}(a,b,c,d) = \sum_{s=a}^{c} \sum_{o=b}^{d} T_{s,o}$$

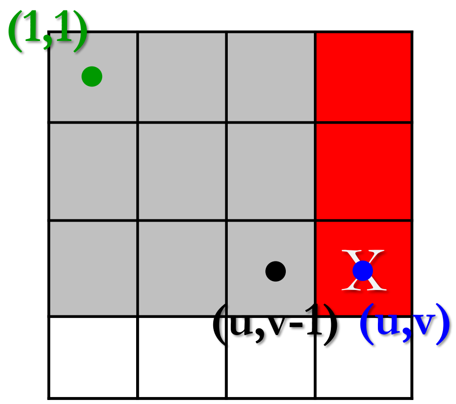
Most ciklust kellene írni i-re, j-re, k-ra, l-re, s-re és o-ra, azaz 6 ciklus lenne egymás belsejében. **Ez sok!**

Segédösszegek

- Az érték függvény újradefiniálása:

  Próbáljunk valami részcélt kitűzni:

  számoljuk ki az (1,1) bal felső, (u,v) jobb alsó sarkú téglalapok értékét!



X= szürke téglalap értéke +
piros téglalap összege

E[u,v] ← X

Segédösszegek

- Az érték függvény újradefiniálása (folytatás):

E_mátrix(E,N,M)

E[1..N,0]:=0

v=1..M

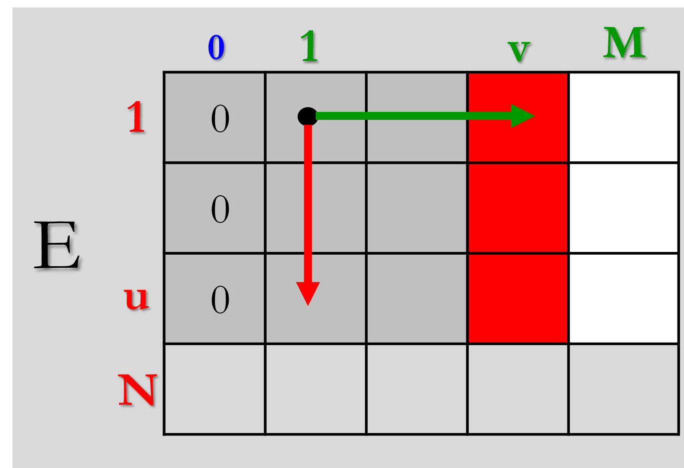
x:=0

u=1..N

x:=x+T[u,v]

E[u,v]:=E[u,v–1]+x

Változó
u,v:**Egész**
x:**Egész**



Tehát E kiszámításához 2, egymásba ágyazott ciklus kell.

Segédösszegek

- Az **érték** függvény újradefiniálása (folytatás):

Definiáljuk **E[u,v]** segítségével az **érték(E,i,j,u,v)**-t!

érték(E,i,j,u,v)

érték:=E[u,v]–E[u,j-1]–E[i-1,v]+E[i-1,j-1]

	…	j-1	j	…	v	…
⋮						
i-1	…	$E_{i-1,j-1}$	$E_{i-1,j}$	…	$E_{i-1,v}$	…
i	…	$E_{i,j-1}$	$E_{i,j}$	…	$E_{i,v}$	
⋮				???		
u	…	$E_{u,j-1}$	$E_{u,j}$		$E_{u,v}$	
⋮						

A módszer neve: **kumulatív összegzés**.

Az **érték függvény** kiszámításához nem kell ciklus → **konstans idejű**. Hozzá **egyszer** kellett az E kiszámítása.

Bemenet: $N,M \in \mathbb{N}$, $T_{1..N,1..M} \in \mathbb{Z}^{N \times M}$

Kimenet: $P,Q,R,S \in \mathbb{N}$

Előfeltétel: –

Utófeltétel: $1 \leq P \leq R \leq N$ és $1 \leq Q \leq S \leq M$ és
$\forall i,j,k,l$ $(1 \leq i \leq k \leq N, 1 \leq j \leq l \leq M)$: érték$(P,Q,R,S) \geq$ érték(i,j,k,l)

Segédösszegek

- A maximális összegű téglalap kiválasztása:

Változó
i,j,k,l,
maxÉrt:**Egész**
E:Tömb[…]

```
E_mátrix(E,N,M)
maxÉrt:=-∞
i=1..N
  j=1..M
    k=i..N
      l=j..M
        I  érték(E,i,j,k,l)>maxÉrt  N
           P,Q,R,S:=i,j,k,l         | —
           maxÉrt:=érték(E,i,j,k,l) |
```

A ciklusban számított **érték konstans** idővel határozható meg!

Összegzés + maximum-kiválasztás

Feladat:

Adott egy N elemű X számsorozat, adjuk meg azt a pontosan K hosszú részintervallumát, amelyben az értékek összege maximális (kezdő szám: Max-adik , összeg: MaxÉrt)!

Alapmegoldás:

Változó
i,j:**Egész**
s:**Egész**

Összegzés:
X[i]+…+X[i+K-1]

```
MaxÉrt:=-∞
i=1..N–K+1
    s:=0
    j=i..i+K–1
        s:=s+X[j]
    s>MaxÉrt
    I: Max:=i; MaxÉrt:=s
    N: —
```

Összegzés + maximum-kiválasztás + ablakozás

Feladat:

Adott egy N elemű X számsorozat, adjuk meg azt a pontosan K hosszú részintervallumát, amelyben az értékek összege maximális (kezdő szám: Max-adik , összeg: MaxÉrt)!

Optimális megoldás:

Egy K hosszú intervallum összege az előző intervallum összegéből egy elem levonásával és egy elem hozzáadásával számolható.

Változó
i:**Egész**
s:**Egész**

```
s:=0
i=1..K
    s:=s+X[i]
Max:=1; Maxért:=s
i=2..N–K+1
    s:=s–x[i–1]+x[i+K–1]
    s>MaxÉrt
    I: Max:=i; MaxÉrt:=s
    N: —
```

Maximum-kiválasztás + keresés

Feladat:

Egy országban az elmúlt N (≥1) napon M (≥1) földrengés volt, ismerjük az egyes földrengések $F_{1..M}$ napsorszámát, időpont szerint növekvő sorrendben. Az is lehet, hogy egy napon több földrengés volt, ekkor a napsorszám ismétlődik. Meg kell adni annak a K napos időszaknak az első napját (Max), amelyen belül a lehető legtöbb (MaxÉrt) földrengés volt!

Alapmegoldás:

Változó
i,j:**Egész**

```
F[M+1]:=N+K; Max:=F[1]; MaxÉrt:=1
i=1..M–1
    j:=i
    j≤M és F[j+1]–F[i]<K
        j:=j+1
    j–i+1>MaxÉrt
    I: Max:=F[i]; MaxÉrt:=j–i+1
    N: —
```

Keresés:
j? : F[j]-F[i]<K és
F[j+1]-F[i]≥K

Maximum-kiválasztás + keresés + ablakozás

Feladat:

Egy országban az elmúlt N (≥1) napon M (≥1) földrengés volt, ismerjük az egyes földrengések $F_{1..M}$ napsorszámát, időpont szerint növekvő sorrendben. Az is lehet, hogy egy napon több földrengés volt, ekkor a napsorszám ismétlődik. Meg kell adni annak a K napos időszaknak az első napját (Max), amelyen belül a lehető legtöbb (MaxÉrt) földrengés volt!

Optimális megoldás (vázlat):

Ha van már egy K napos [Kezdet,Vég] intervallumunk, akkor a Kezdet növelésekor az intervallum vége (Vég) folyamatosan növelhető.

Az_első_időszak_megkeresése
Ablakos_maximum_kiválasztás

Maximum-kiválasztás + keresés + ablakozás

Optimális megoldás:

Változó
kezdet,
vég:**Egész**

```
F[M+1]:=N+K; kezdet:=1; vég:=1
vég≤M és F[vég+1]–F[kezdet]<K
    vég:=vég+1
Max:=F[kezdet]; MaxÉrt:=vég–kezdet+1
kezdet=2..M–1
    [vég:=vég]
    vég≤M és F[vég+1]–F[kezdet]<K
        vég:=vég+1
    vég–kezdet+1>MaxÉrt
    I: Max:=F[kezdet]; MaxÉrt:=vég–kezdet+1
    N: —
```

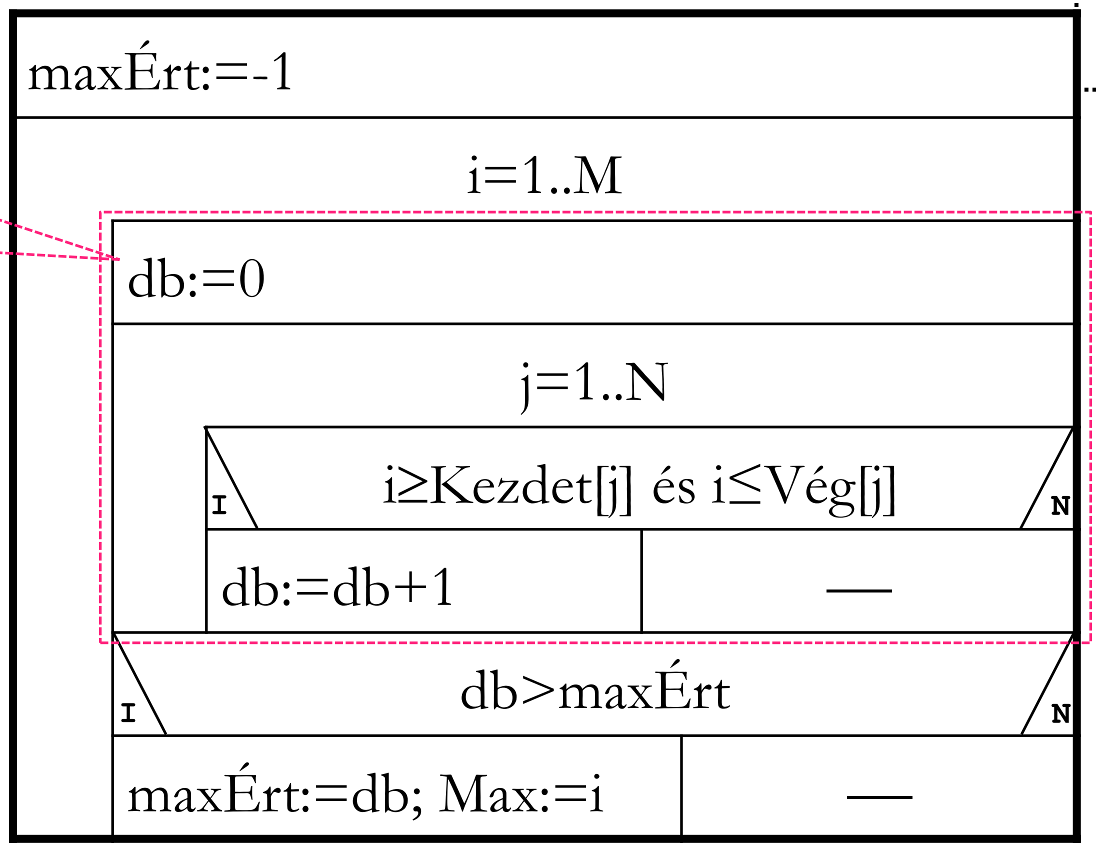
Maximum-kiválasztás + megszámlálás

Feladat: Adott N intervallum, kezdő- és végpontjaik (Kezdet,Vég) 1 és M közötti számok. (N,M≥1) Adjunk meg egy értéket (Max), amely a legtöbb intervallumban benne van!

Változó
i,j:**Egész**
maxÉrt:**Egész**

Alapmegoldás:

Megszámolás: az i hány intervallumba esik?



Maximum-kiválasztás + megszámlálás + változásfigyelés

Optimális megoldás:

Legyen a darab[i] jelentése, az i **érték mennyivel több** intervallumban szerepel, mint az i–1.

Változó
i,db,
maxÉrt:**Egész**
darab:Tömb[…]

darab[1..M+1]:=0

i=1..N

- darab[Kezdet[i]]:=darab[Kezdet[i]]+1
- darab[Vég[i]+1]:=darab[Vég[i]+1]–1

db:=darab[1]; Max:=1; maxÉrt:=db

i=2..M

- db:=db+darab[i]
- db>maxÉrt

I	N
Max:=i; maxÉrt:=db	—

Keresés

Feladat:

Adott egy növekvő N (≥2) elemű X számsorozat. Jelöljük ki két elemét (A, B), amelyek összege pontosan Z!

Alapmegoldás:

Változó
i,j:**Egész**

i:=1; j:=2

i<N és X[i]+X[j]≠Z

I	j<N	N
j:=j+1		i:=i+1; j:=i+1

Van:=i<N

I	Van	N
A:=i; B:=j		—

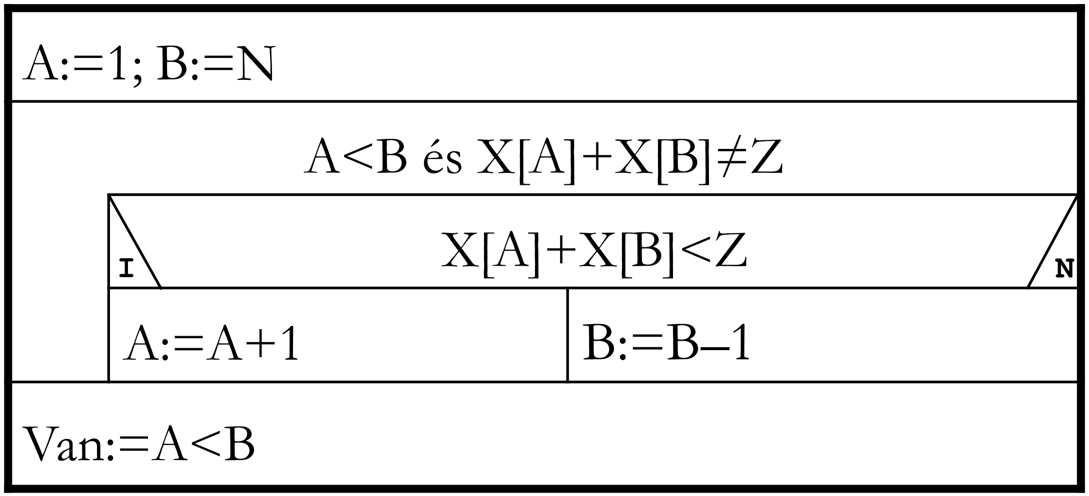
Keresés + intervallumszűkítés

Feladat:

Adott egy növekvő N (≥2) elemű X számsorozat. Jelöljük ki két elemét (A, B), amelyek összege pontosan Z!

Optimális megoldás:

Ha az első és utolsó elem összege kisebb Z-nél, akkor az első biztosan nem megoldás. Ha nagyobb, akkor az utolsó biztosan nem megoldás.



Maximum-kiválasztás + keresés

Feladat:

Adott egy N elemű X sorozat, amely 1 és M közötti értékeket tartalmaz. Adjuk meg azt az A értéket, amely két előfordulása a lehető legközelebb van egymáshoz ha van egyáltalán ismétlődő érték (Van)! (N≥2,M≥1)

Alapmegoldás:

```
minTáv:=N
i=1..N–1
    j:=i+1
    j≤N és X[i]≠X[j]
        j:=j+1
    j≤N és j–i<minTáv
        I: minTáv:=j–i; min:=i
        N: —
Van:=minTáv<N
Van
    I: A:=X[min]
    N: —
```

Keresés:
X[i] következő
ismétlődése

Változó
i,j,minTáv,
min:**Egész**

Maximum-kiválasztás + keresés + intervallumkezdet megőrzése

Feladat:

Adott egy N elemű X sorozat, amely 1 és M közötti értékeket tartalmaz. Adjuk meg azt az A értéket, amely két előfordulása a lehető legközelebb van egymáshoz, ha van egyáltalán ismétlődő érték (Van)! (N≥2,M≥1)

Optimális megoldás:

Minden értékhez tároljuk az utolsó előfordulása helyét!

Változó
i,minTáv,
min:**Egész**
ut:Tömb[…]

minTáv:=N; ut[1..M]:=-N

i=1..N

I	i–ut[X[i]]<minTáv	N
minTáv:=i–ut[X[i]]; min:=ut[X[i]]		–

ut[X[i]]:=i

Van:=minTáv<N

I	Van	N
A:=X[min]		–

Rendezés

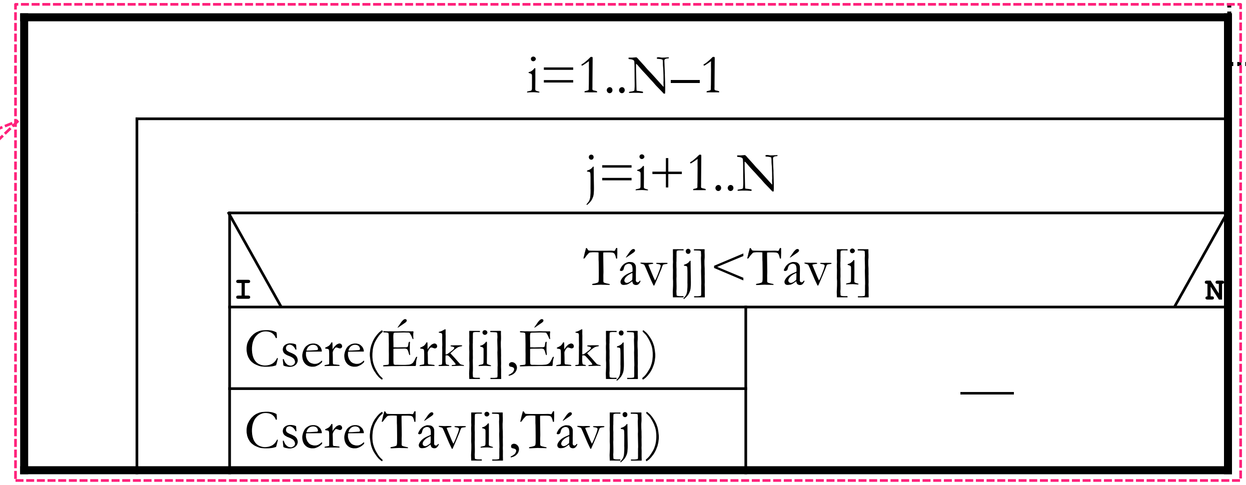
Feladat:

Egy rendezvényen N vendég vesz részt. Érkezési sorrendben ismerjük mindegyik érkezési (Érk) és távozási (Táv) idejét, mindkettő 1 és M közötti egész szám. Sem érkezni, sem távozni nem akart két vendég egyszerre. Adjuk meg a vendégeket távozási idő szerinti sorrendben!

Alapmegoldás:

Változó
i,j:**Egész**

Egyszerű cserés rendezés



Kiválogatás +
intervallumkezdet megőrzése, párok indexelése

Optimális megoldás:

Egy intervallum végekkel indexelt tömbbe tegyük bele az intervallum kezdeteket!

Változó
i,j,db:**Egész**
kezd:Tömb[…]

```
kezd[1..M]:=0
i=1..N
    kezd[Táv[i]]:=Érk[i]
db:=0
i=1..M
    kezd[i]>0
    I:
        db:=db+1
        Érk[db]:=kezd[i]; Táv[db]:=i
    N:
        –
```

Rekurzió

Klasszikus példák:

- Faktoriális

$$n! = \begin{cases} n*(n-1)! & ha\ \ n>0 \\ 1 & ha\ \ n=0 \end{cases}$$

- Fibonacci-számok

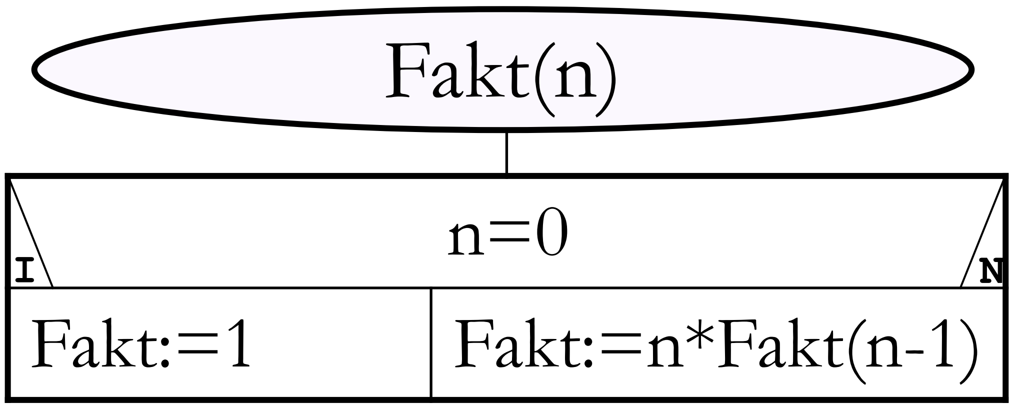
$$Fib(n) = \begin{cases} 0 & ha\ \ n=0 \\ 1 & ha\ \ n=1 \\ Fib(n-1)+Fib(n-2) & ha\ \ n>1 \end{cases}$$

A rekurzió lényege: önhivatkozás

Rekurzív specifikáció és algoritmus

Faktoriális:

$$n! = \begin{cases} n*(n-1)! & ha\ \ n>0 \\ 1 & ha\ \ n=0 \end{cases}$$



Itt egy 2-alternatívájú függvényt kell algoritmizálni, ami egy „2-irányú” elágazással történik.

Rekurzív specifikáció és algoritmus

Fibonacci-számok:

$$Fib(n) = \begin{cases} 0 & ha\ \ n = 0 \\ 1 & ha\ \ n = 1 \\ Fib(n-1) + Fib(n-2) & ha\ \ n > 1 \end{cases}$$

Fib(n)

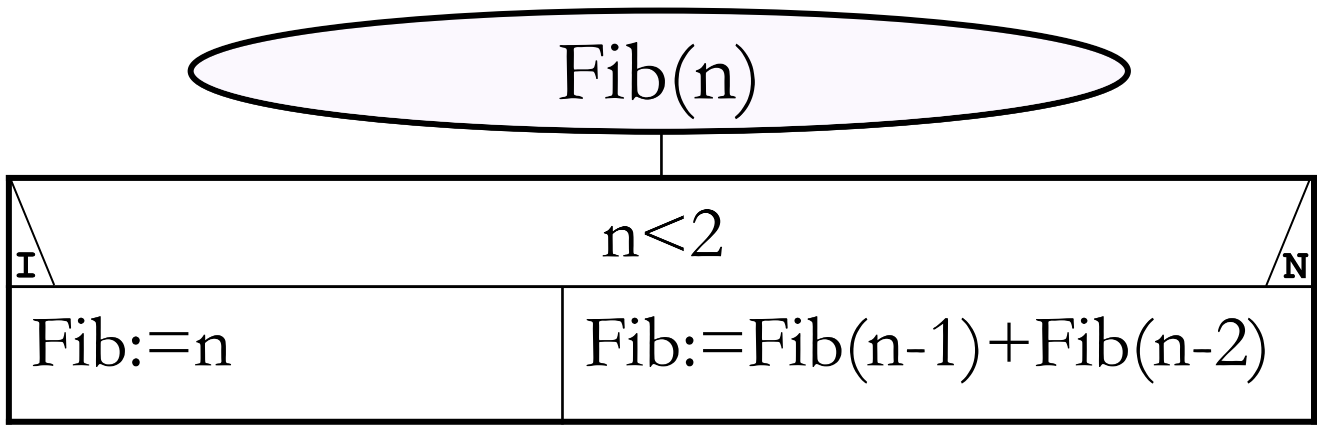
n=0	n=1	n>1
Fib:=0	Fib:=1	Fib:=Fib(n-1)+Fib(n-2)

Háromirányú elágazás a megoldás.

Rekurzív specifikáció és algoritmus

Fibonacci-számok:

$$Fib(n)=\begin{cases}0 & ha\ \ n=0\\ 1 & ha\ \ n=1\\ Fib(n-1)+Fib(n-2) & ha\ \ n>1\end{cases}$$



Kétirányú elágazássá alakított megoldás.

Problémák a rekurzióval

Hely: nagyra dagadt veremméret.

Idő: a vermelés adminisztrációs többletterhe, a többszörösen ismétlődő hívások.

Példa: **Fibonacci-számok esetén**

r(i):=az i. Fibonacci-szám kiszámításához szükséges hívások száma

r(0):=1, r(1):=1, r(i):=r(i–1)+r(i–2)+1 (i>1)

$$Fib(n) = \begin{cases} 0 & ha\ n = 0 \\ 1 & ha\ n = 1 \\ Fib(n-1) + Fib(n-2) & ha\ n > 1 \end{cases}$$

Állítás:

a) r(i)=F(i+1)+F(i)+F(i–1)–1 (i>1),

b) r(i)=2*F(i+1)–1,

ahol F(i)=az i. Fibonacci-szám.

c) r(i)=$\Theta(c^i)$, azaz exponenciális műveletigényű.

Rekurzió és iteráció

Korlátos memóriájú függvények:

Ha egy rekurzív függvény minden értéke valamely korábban kiszámolható értékből számolható, akkor némi memória felhasználással elkészíthető a rekurziómentes változat, amelyben az egyes függvényértékeknek megfeleltetünk egy F[0..N] vektort.

A függvény általános formája:

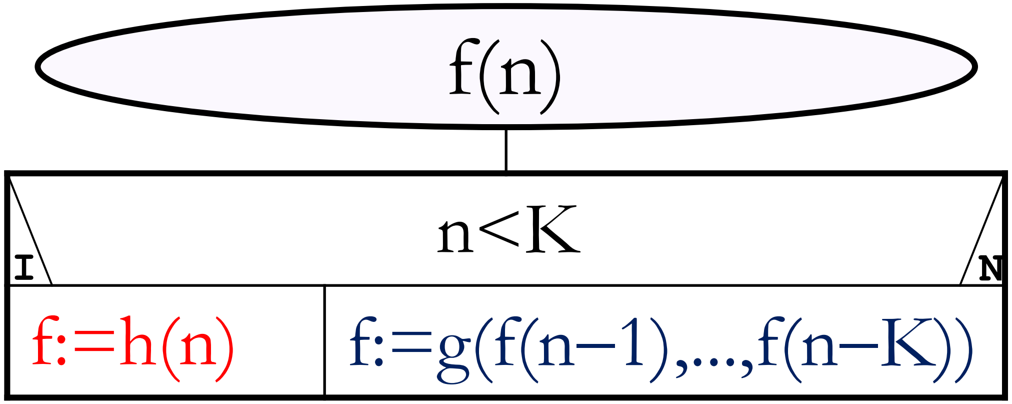
$$f(n) = \begin{cases} g\big(f(n-1), f(n-2), \ldots, f(n-K)\big) & \text{ha} \quad n \geq K \\ h(n) & \text{ha} \quad 0 \leq n < K \end{cases}$$

Rekurzió és iteráció

Korlátos memóriájú függvények:

Rekurzív változat:

$$f(n)=\begin{cases} g\big(f(n-1),f(n-2),\ldots,f(n-K)\big) & \text{ha} \quad n \geq K \\ h(n) & \text{ha} \quad 0 \leq n < K \end{cases}$$



Rekurzió és iteráció

Korlátos memóriájú függvények:

Iteratív (ciklusos) változat:

$$f(n)=\begin{cases} g\big(f(n-1),f(n-2),\ldots,f(n-K)\big) & \text{ha} \quad n\ge K \\ h(n) & \text{ha} \quad 0\le n<K \end{cases}$$

f(n)

i=0..K–1

F[i]:=h(i)

i=K..n

F[i]:=g(F[i–1],...,F[i–K])

f:=F[n]

Változó
F:**Tömb**[0..n:TH]

Rekurzió és iteráció

Ez így természetesen nem hatékony tárolás, hiszen a rekurzív formulából látszik, hogy minden értékhez csak az őt megelőző K értékre van szükség.

A hatékony megoldásban az alábbi értékadást kell átalakítani:

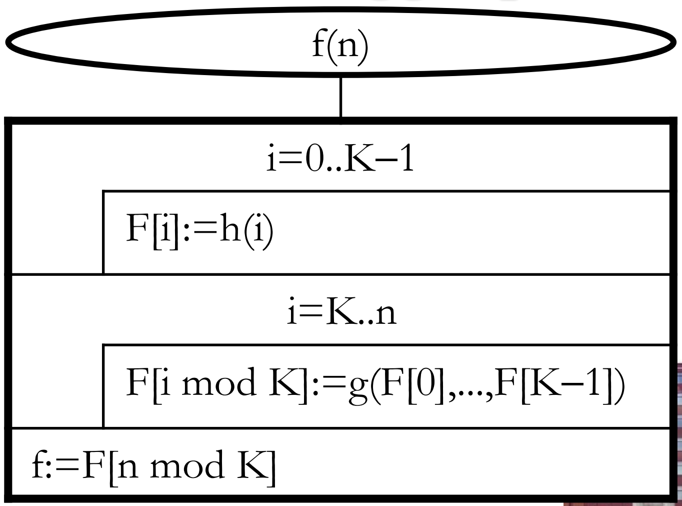
```
F[i]:=g(F[i-1],...,F[i-K])
```

Lehet pl. így, ha a `g()` függvény kiszámítása nem függ a paraméterek sorrendjétől :

```
F[i mod K]:=g(F[0],...,F[K-1]).
```

Ekkor elegendő: F[0..K–1] tömb.

Az eredmény az F[n mod K]-ban képződik.



Rekurzió és iteráció

Példa: Fibonacci-számok$_{iteratív}$

$$Fib(n) = \begin{cases} 0 & ha\ n = 0 \\ 1 & ha\ n = 1 \\ Fib(n-1) + Fib(n-2) & ha\ n > 1 \end{cases}$$

Alapmegoldás:

Fib(n)

F[0]:=0; F[1]:=1

i=2..n

F[i]:=F[i–1]+F[i–2]

Fib:=F[n]

Változó
i:**Egész**
F:Tömb[0..n:**Egész**]

Rekurzió és iteráció

Példa: Fibonacci-számok$_{iteratív}$

$$Fib(n) = \begin{cases} 0 & ha\ n = 0 \\ 1 & ha\ n = 1 \\ Fib(n-1) + Fib(n-2) & ha\ n > 1 \end{cases}$$

Helytakarékos megoldás (K=2):

Fib(n)

F[0]:=0; F[1]:=1

i=2..n

F[i mod 2]:=F[0]+F[1]

Fib:=F[n mod 2]

Változó

i:**Egész**

F:Tömb[0..1:**Egész**]

2022.11.29.

Rekurzió memorizálással

Többszörös hívás elkerülése:

Amit már kiszámoltunk egyszer, azt ne számoljuk újra! Tároljuk a már kiszámolt értékeket, és ha újra szükségünk van rájuk, használjuk fel őket!

Példa: **Fibonacci-számok esetén**

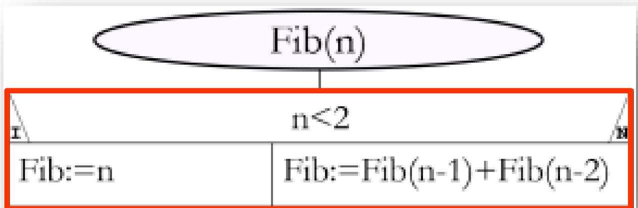
A megoldásban F[i]≥0 jelentse, ha már kiszámoltuk az i-edik Fibonacci-számot.

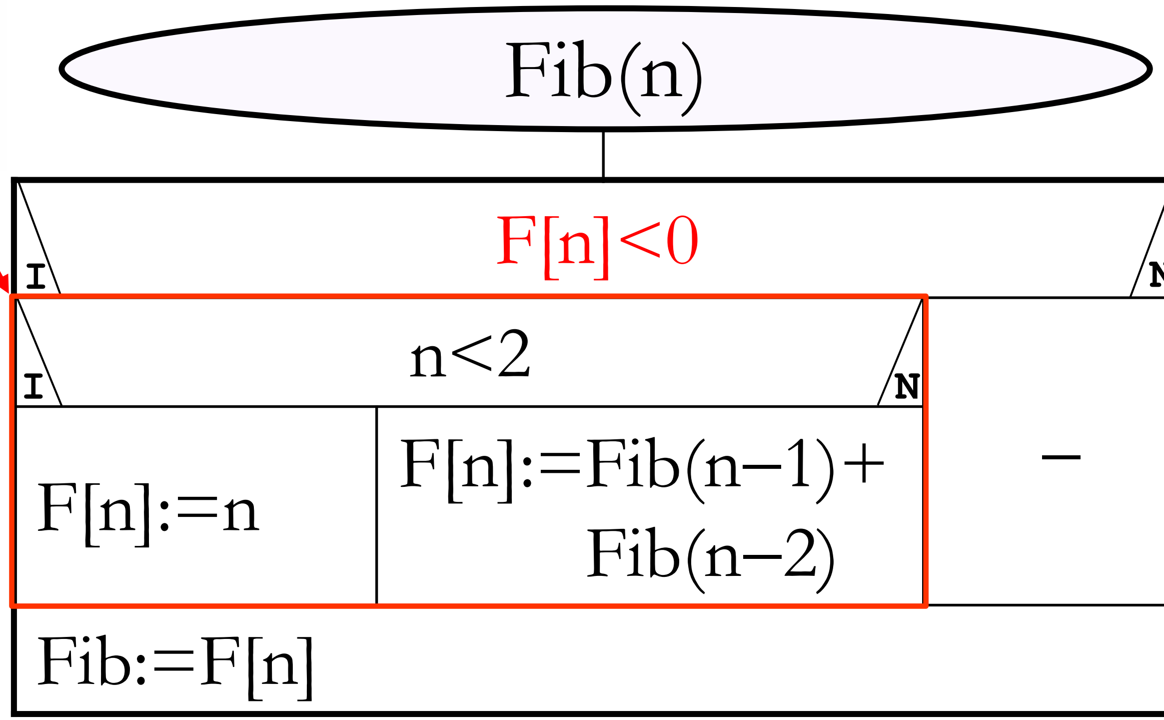
Változó
  i:**Egész**
  F:**Tömb**[0..N:**Egész**]

```
i=0..N
    F[i]:=-1
F[N]:=Fib(N)
```

Rekurzió memorizálással

Algoritmus (folytatás):





Közvetett rekurzió

Feladat:

Döntsük el egy számról, hogy páros-e, ha nincs maradék-számítás műveletünk!

Megoldás:

Páros(n)

n=0	n=1	
Páros:=Igaz	Páros:=Hamis	Páros:=Páratlan(n−1)

Mivel párost páratlan előz meg.

Páratlan(n)

n=0	n=1	
Páratlan:=Hamis	Páratlan:=Igaz	Páratlan:=Páros(n−1)

Mivel páratlant páros előz meg.

Közvetlen rekurzió

Feladat:

Döntsük el egy számról, hogy páros-e, ha nincs maradék-számítás műveletünk!

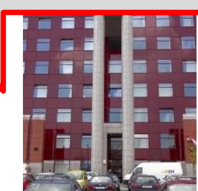
A két – közvetetten – rekurzív eljárás most összevonható:

Megoldás:

Páros(n)

n=0	n=1	
Páros:=Igaz	Páros:=Hamis	Páros:=Páros(n−2)

Mivel párost nem páros, és nem párost páros előz meg.



2022.11.29.

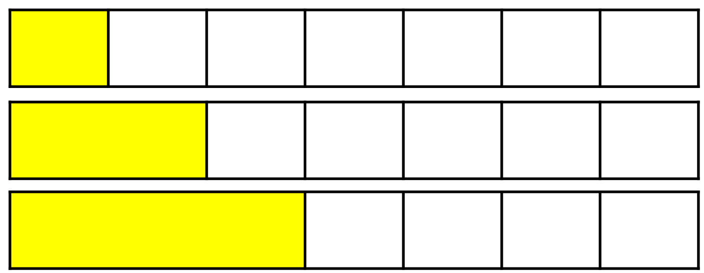
Közvetlen rekurzió – járdakövezés

Feladat:

Számítsuk ki, hogy hányféleképpen lehet egy 1×n egység méretű járdát kikövezni 1×1, 1×2 és 1×3 méretű lapokkal!

Az első helyre tehetünk:

- 1×1-es lapot:
- 1×2-es lapot:
- 1×3-as lapot:

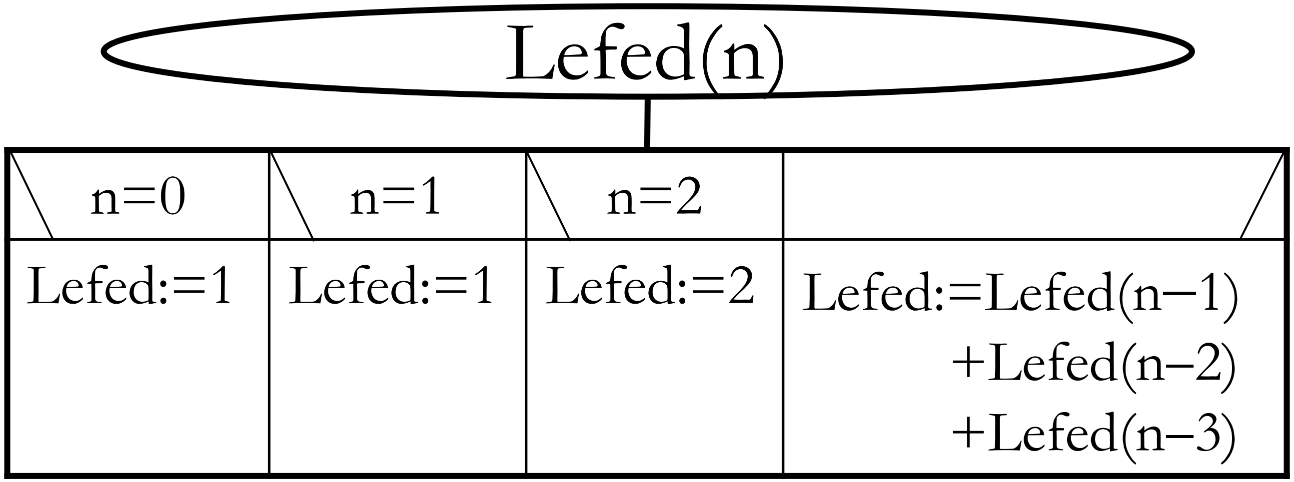


Az első esetben n–1, a másodikban n–2-t, a harmadikban pedig n–3 cellát kell még lefednünk. Azaz az n cella lefedéseinek száma:

Lefed(n)=Lefed(n–1)+Lefed(n–2)+Lefed(n–3), ha n>2.

Közvetlen rekurzió – járdakövezés

Megoldás:



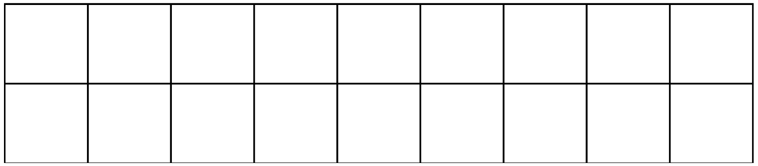
Sokszoros hívás kiküszöbölése:

- vagy **memorizálás**sal,
- vagy **iteratív** (ciklusos) implementálással!

Közvetett rekurzió – járdakövezés

Feladat:

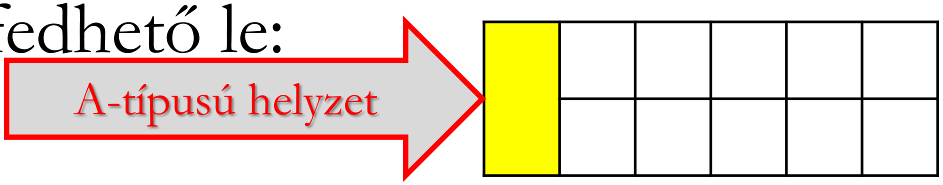
Számítsuk ki, hogy hányféleképpen lehet egy 2×n egység méretű járdát kikövezni 1×2 és 1×3 méretű lapokkal!



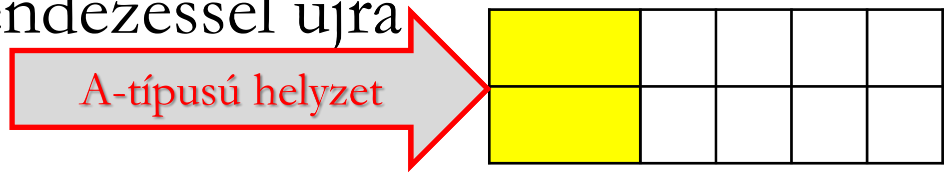
Közvetett rekurzió – járdakövezés

Megoldás:

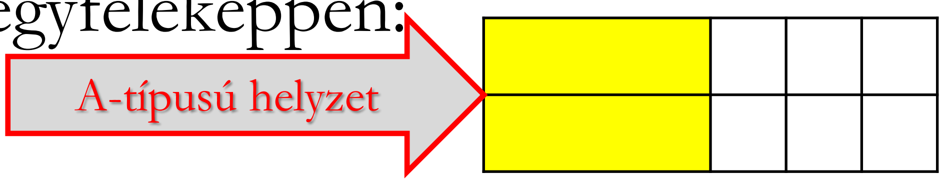
Az első oszlop egyféleképpen fedhető le:



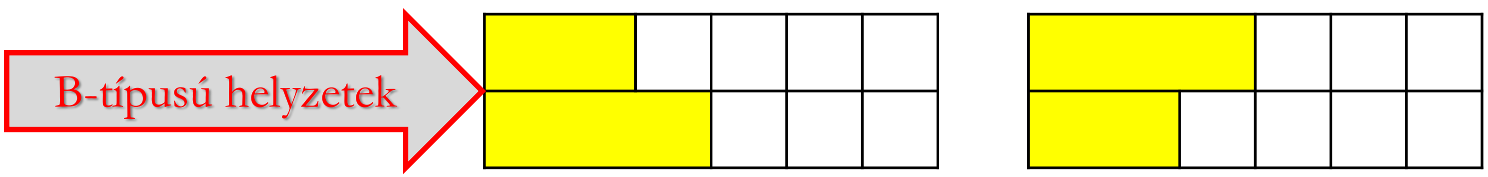
Az első két oszlop további elrendezéssel újra egyféleképpen fedhető le:



Az első három oszlop … újra egyféleképpen:



Sajnos ez is előfordulhat:

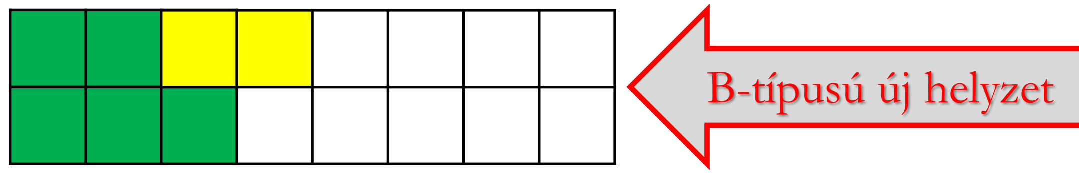


2022.11.29.

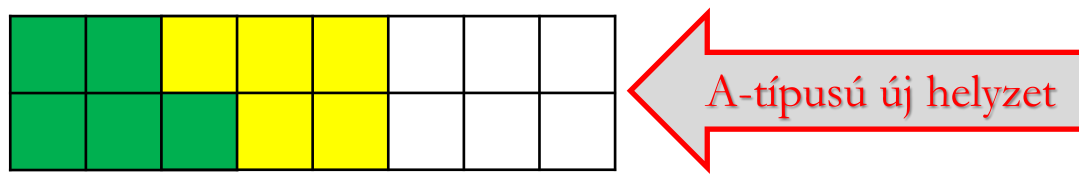
Közvetett rekurzió – járdakövezés

A B-típusú helyzetű járda háromféleképpen folytatható:

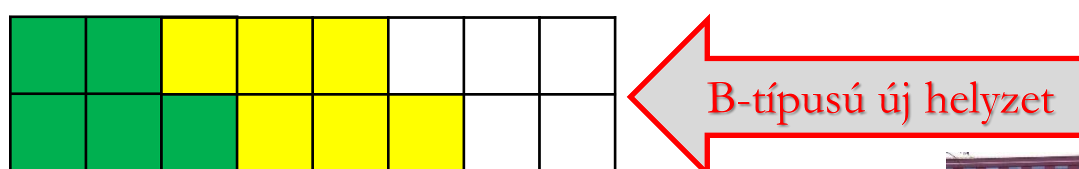
- Ha fölülre „kettes”-t teszünk:



- Ha fölülre „hármast” és alulra „kettes”-t teszünk:

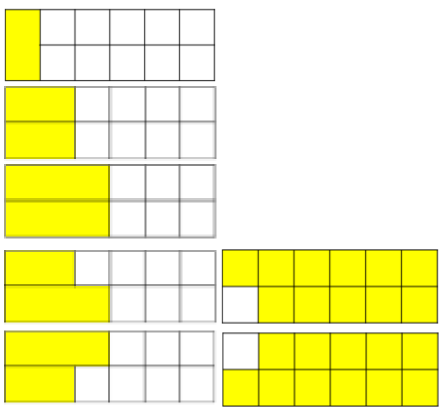


- Ha fölülre „hármast” és alulra „hármas”-t teszünk:

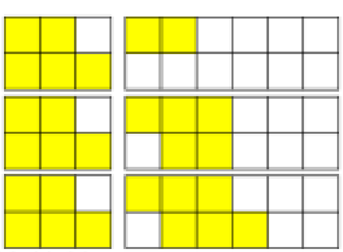


Közvetett rekurzió – járdakövezés

Jelölje A(n) a megoldás értékét 2×n egység méretű járda esetén! Jelölje B(n) a megoldás értékét 2×n egység méretű járda esetén, ha az egyik baloldali sarok nincs lefedve!



$$A(n)=\begin{cases}1 & \text{ha} \quad n=1\\ 2 & \text{ha} \quad n=2\\ 4 & \text{ha} \quad n=3\\ A(n-1)+A(n-2)+A(n-3)+2*B(n-2) & \text{ha} \quad n>3\end{cases}$$



$$B(n)=\begin{cases}0 & \text{ha} \quad n=1\\ 0 & \text{ha} \quad n=2\\ 1 & \text{ha} \quad n=3\\ B(n-1)+A(n-3)+B(n-3) & \text{ha} \quad n>3\end{cases}$$

Közvetett rekurzió – járdakövezés

$$A(n)=\begin{cases}1 & \text{ha} \quad n=1\\ 2 & \text{ha} \quad n=2\\ 4 & \text{ha} \quad n=3\\ A(n-1)+A(n-2)+A(n-3)+2*B(n-2) & \text{ha} \quad n>3\end{cases}$$

A(n)

n=1	n=2	n=3	
A:=1	A:=2	A:=4	A:=A(n–1)+A(n–2)+A(n–3)+ 2*B(n-2)

$$B(n)=\begin{cases}0 & \text{ha} \quad n=1\\ 0 & \text{ha} \quad n=2\\ 1 & \text{ha} \quad n=3\\ A(n-3)+B(n-1)+B(n-3) & \text{ha} \quad n>3\end{cases}$$

B(n)

n<3	n=3	
B:=0	B:=1	B:=A(n–3)+B(n–1)+B(n–3)

Rekurzív eljárás

A rekurzív eljárások nem mindig alakíthatók át egyszerűen táblázatkitöltéssé, az alábbi feladat nemrekurzív megoldása sokkal nehezebb lehet.

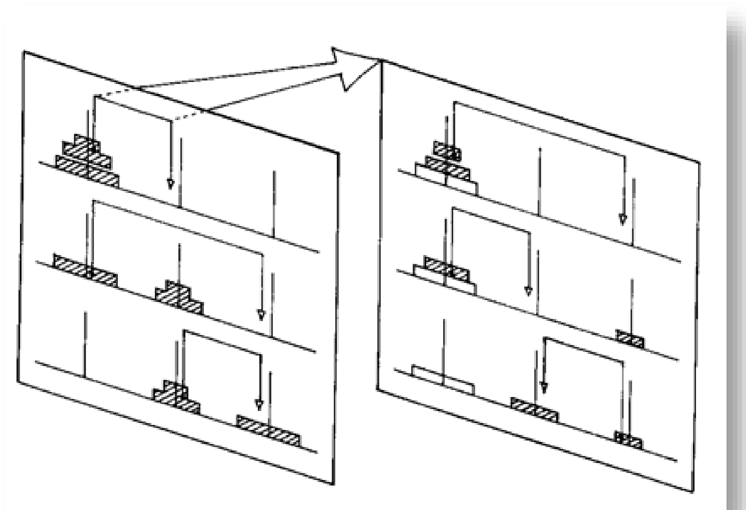
Hanoi tornyai:

Adott 3 rudacska. Az elsőn egyre csökkenő sugarú korongok vannak. Az a feladat, hogy tegyük át a harmadik rudacskára a korongokat egyenként úgy, hogy az átpakolás közben és természetesen a végén is minden egyes korongon csak nála kisebb lehet. Az átpakoláshoz lehet segítségül felhasználni a középső rudacskát.

Rekurzív eljárás

Hanoi tornyai:

- „N−1 darabot 1-ről 2-re”
- „Legalsót (N=1) 1-ről 3-ra”
- „N−1 darabot 2-ről 3-ra”



Hanoi(n,ról,át,ra)

I	N>1	N
Hanoi(n−1,ról,ra,át)		
Ki: n,ról,ra		Ki: n,ról,ra
Hanoi(n−1,át,ról,ra)		

Programozási tételek rekurzívan

Sorozatszámítás (összegzés):

A sorozatszámítás tétel egy egyszerű rekurziót tartalmazott, ahol minden kiszámolt érték az előző egyetlen értéktől függött:

$$F(X_{1..n}) := \begin{cases} F_0 & , n = 0 \\ f(F(X_{1..n-1}), X_n) & , n > 0 \end{cases}$$

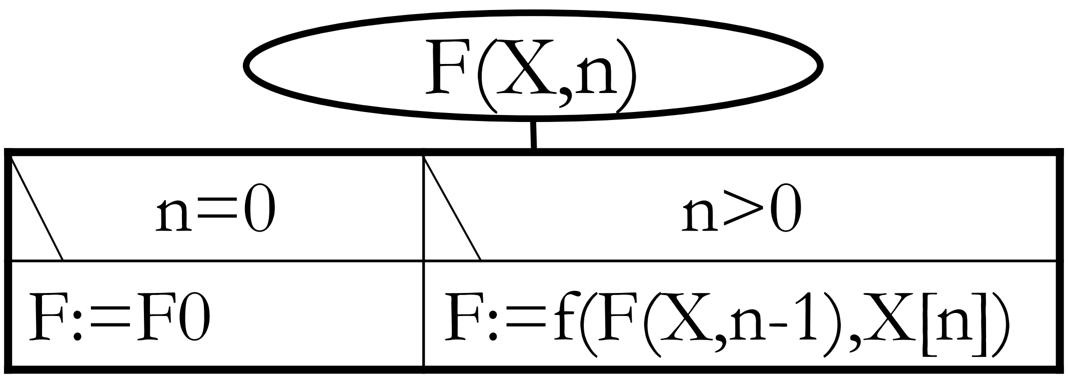
Programozási tételek rekurzívan

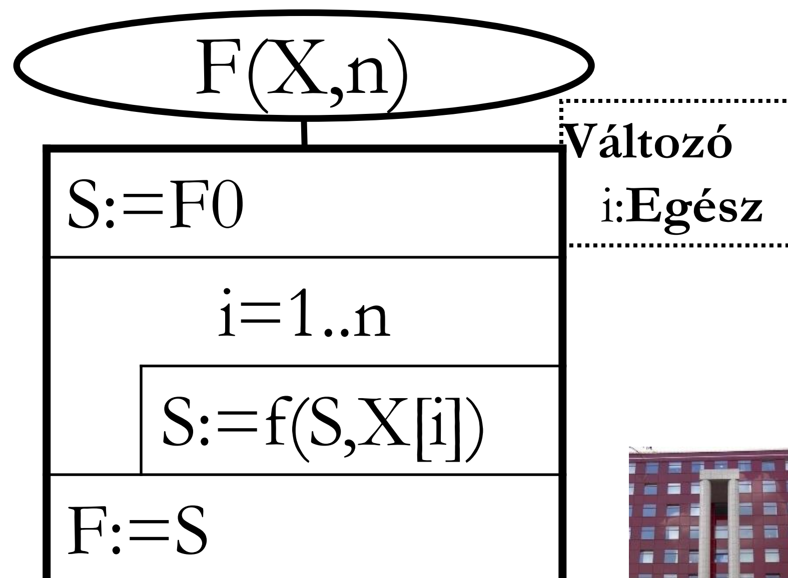
$$F(X_{1..n}) := \begin{cases} F_0\,, & n = 0 \\ f(F(X_{1..n-1}), X_n)\,, & n > 0 \end{cases}$$

Sorozatszámítás (összegzés):

A sorozatszámítás tétel egy egyszerű rekurziót tartalmazott, ahol minden kiszámolt érték az előző egyetlen értéktől függött:

$$F(X, n) := \begin{cases} F_0 & , n = 0 \\ f(F(X, n-1), X_n) & , n > 0 \end{cases}$$





Programozási tételek rekurzívan

Maximum-kiválasztás:

A maximum-kiválasztás tétel rekurzívan ugyanezen az elven fogalmazható meg:

$$\mathrm{Maximum}(X, n) := \begin{cases} X_1 & , n = 1 \\ \max(\mathrm{Maximum}(X, n-1), X_n) & , n > 1 \end{cases}$$

Maximum(X,n)

n=1	n>1
Maximum:=X[1]	Maximum:= max(Maximum(X,n-1),X[n])

Programozási tételek rekurzívan

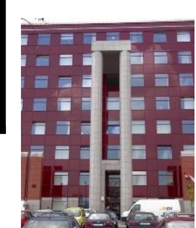
Keresés:

A keresés tétel is ugyanezen az elven fogalmazható meg rekurzívan, de már háromirányú elágazással:

$$\text{Keresés}(X,n) := \begin{cases} (\text{hamis}, -) & , n = 0 \\ (\text{igaz}, n) & , T(X_n) \\ \text{Keresés}(X, n-1) & , \text{egyébként} \end{cases}$$

Keresés(X,n)

n=0	T(X[n])	
Keresés:= (hamis,-)	Keresés:= (igaz,n)	Keresés:= Keresés(X,n−1)



Visszatekintés

- Programtranszformációk
- Hatékony algoritmikus technikák
  - Segédösszegek számítása
  - Ablakozás (2-féleképpen)
  - Változásfigyelés
  - Intervallum-manipulációk (3-féleképpen)
- Rekurzió
  - Rekurzió és iteráció
  - Programozási tételek rekurzívan