ELTE | IK

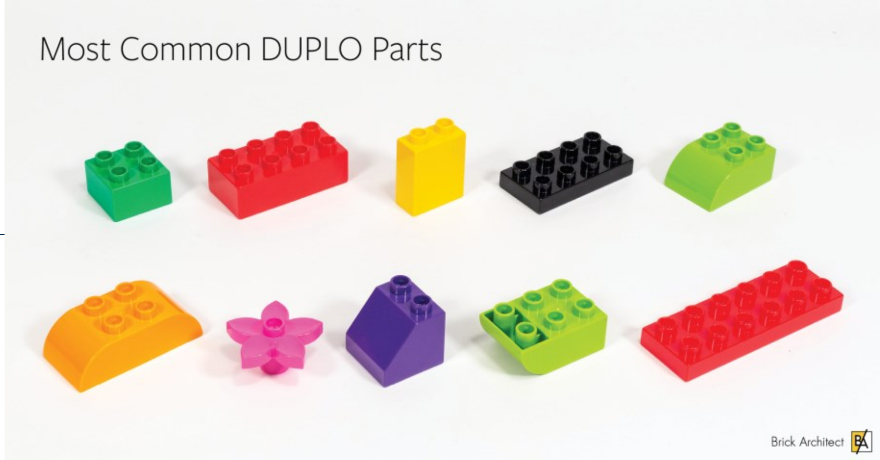
PROGRAMOZÁS

Rekurzió

Horváth Győző

Programozási minták

1. Összegzés
2. Megszámolás
3. Maximumkiválasztás
    a. Minimumkiválasztás
4. Feltételes maximumkeresés
5. Keresés
6. Eldöntés
    a. Mind eldöntés
7. Kiválasztás
8. Másolás
9. Kiválogatás



Rekurzió

Klasszikus példák rekurzióra

- 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

Faktoriális:

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

```
Be: n∈N
Ki: f∈N


Ef: -
Uf: f=n!
```

Rekurzív specifikáció

Faktoriális:

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

```
Be: n∈N
Ki: f∈N
Fv: fakt:N->N, fakt(n)={n*fakt(n-1), ha n>0;
                        1           , ha n=0}
Ef: -
Uf: f=fakt(n)
```

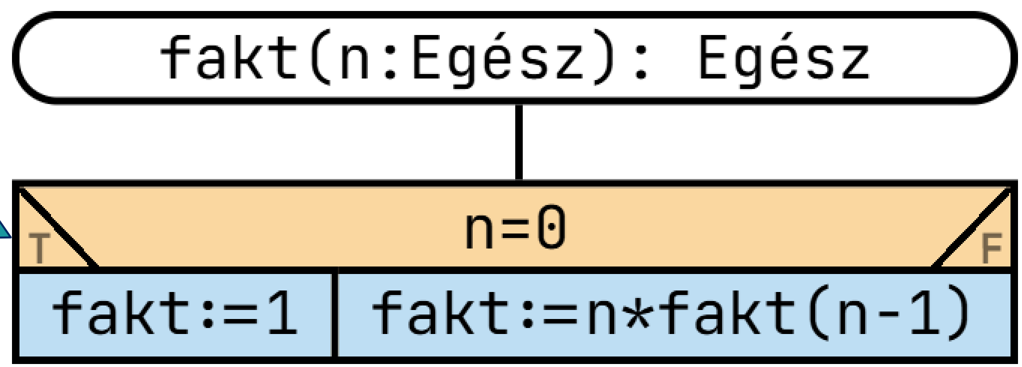
Rekurzív specifikáció és algoritmus

Faktoriális:

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

```
Be: n∈N
Ki: f∈N
Fv: fakt:N->N, fakt(n)={n*fakt(n-1), ha n>0;
                        1          , ha n=0}
Ef: -
Uf: f=fakt(n)
```

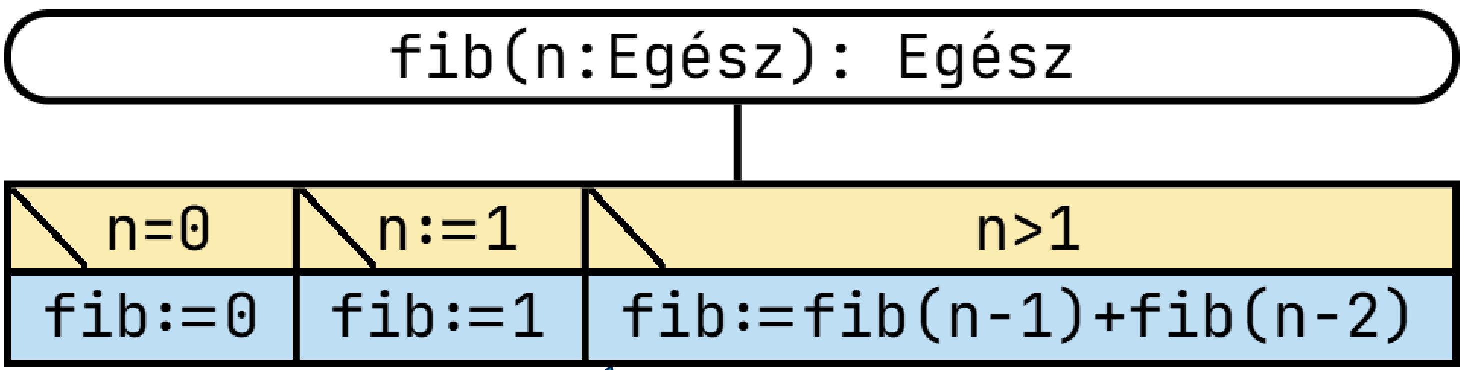
Itt egy 2-alternatívájú függvényt kell algoritmizálni, ami egy „2-irányú” elágazással történik.



Rekurzív specifikáció és algoritmus

Fibonacci-számok:

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

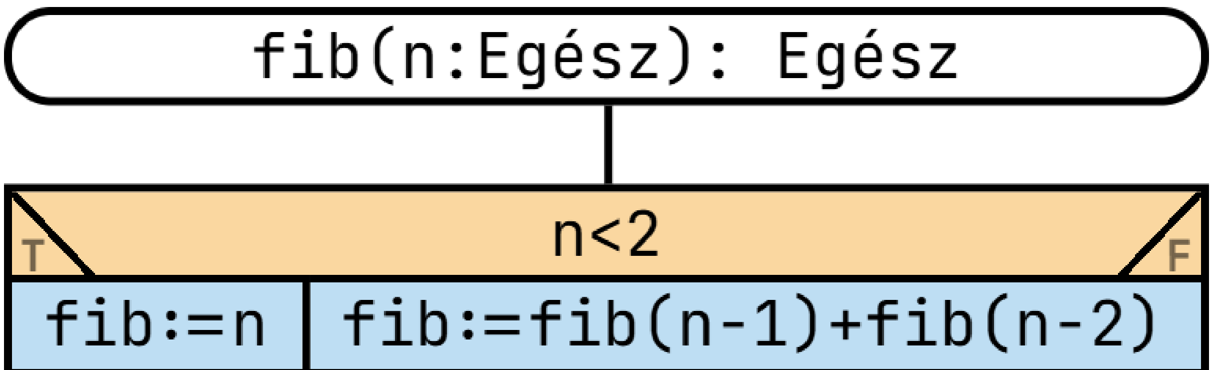


Háromirányú elágazás a megoldás

Rekurzív specifikáció és algoritmus

Fibonacci-számok:

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

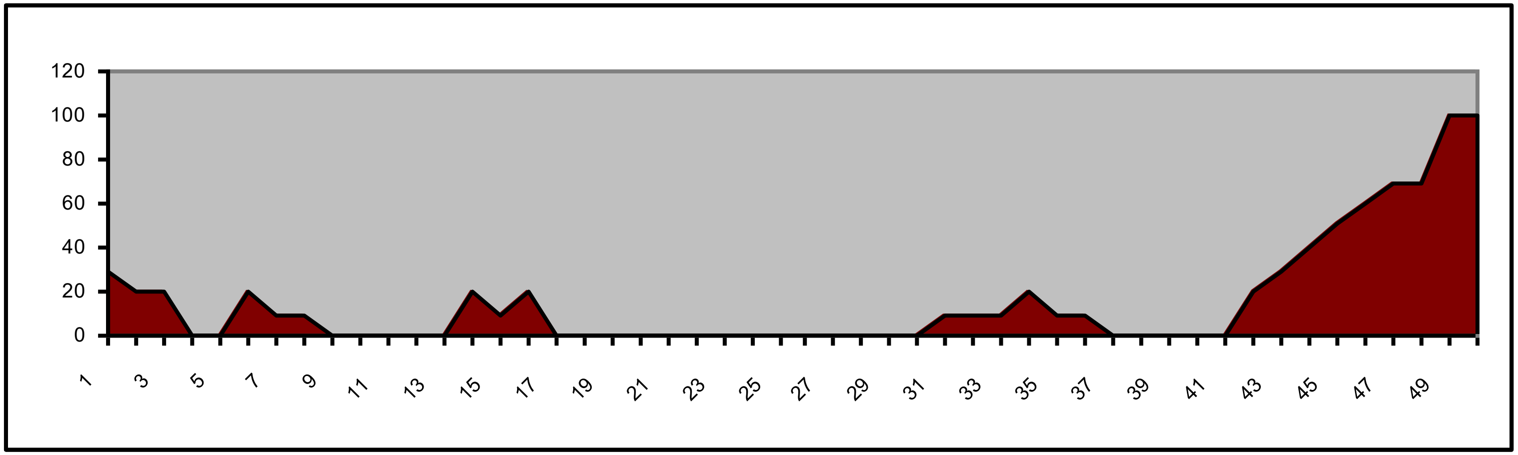


Kétirányú elágazássá alakított megoldás

Intervallumos példák

Feladat:

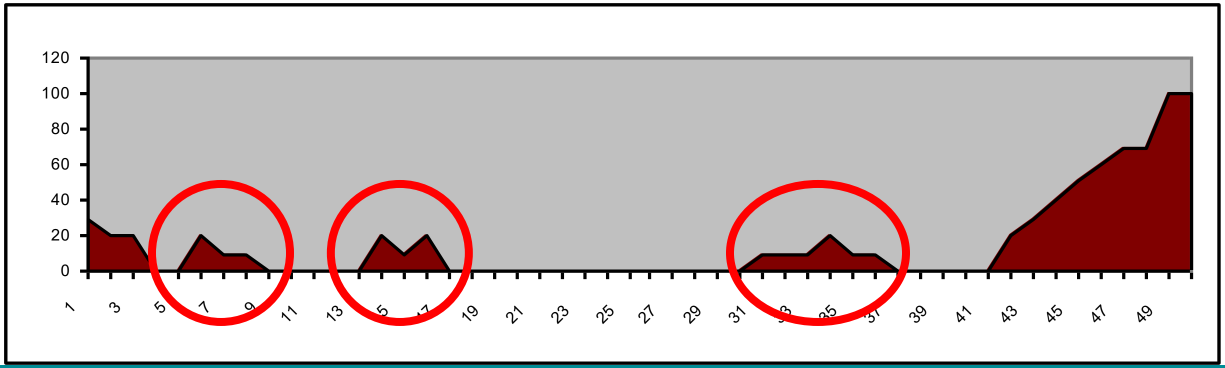
Egy repülőgéppel Európából Amerikába repültünk. Az út során bizonyos kilométerenként mértük a felszín tengerszint feletti magasságát (≥0). 0 magasságot ott mértünk, ahol tenger van, >0-t pedig ott, ahol szárazföld. Adjuk meg a legszélesebb szigetet!



Mi számít szigetnek?

1. Szélén kontinens

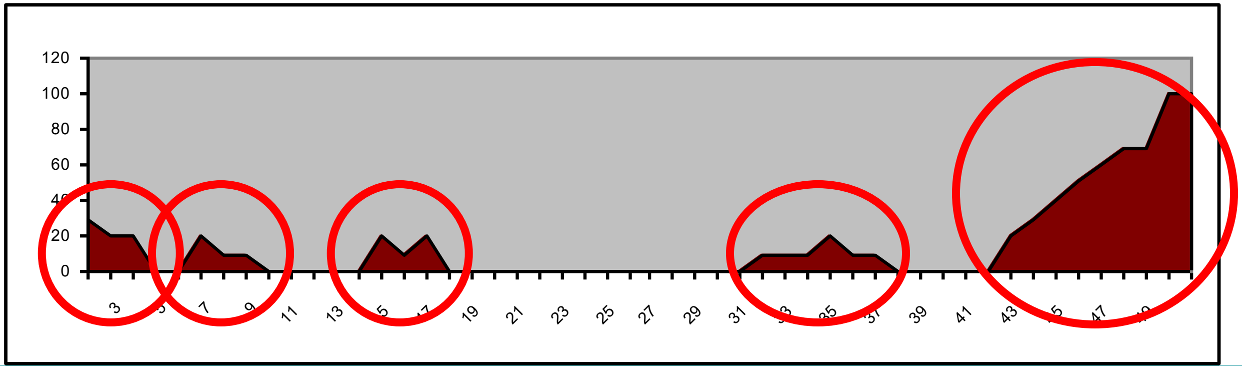
```
Fv: szigetkezdet:N->L,
    szigetkezdet(i)={hamis, ha i=1;
                     mag[i-1]=0 és mag[i]>0 egyébként}
Fv: szigetvég:N->L,
    szigetvég(i)=   {hamis, ha i=n;
                     mag[i+1]=0 és mag[i]>0 egyébként}
```



Mi számít szigetnek?

2. Szélén sziget

```
Fv: szigetkezdet:N->L,
    szigetkezdet(i)={mag[i]>0, ha i=1;
                     mag[i-1]=0 és mag[i]>0 egyébként}
Fv: szigetvég:N->L,
    szigetvég(i)=   {mag[i]>0, ha i=n;
                     mag[i+1]=0 és mag[i]>0 egyébként}
```

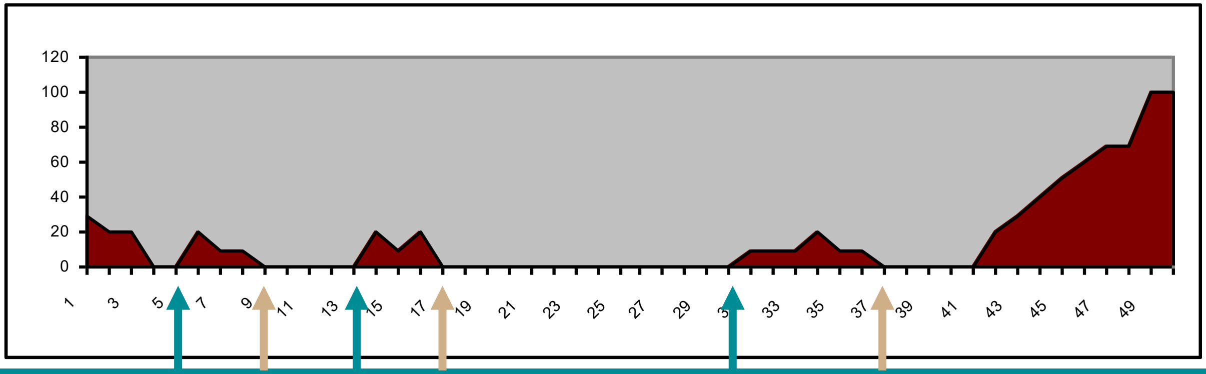


Programozási mintákkal

1. Válogassuk ki azokat a szigetkezdeteket, amelyeknek van vége is! (összevonható)
2. Másolással határozzuk meg hozzájuk a szigetvégeket! (összevonható)
3. Másolással határozzuk meg a távolságokat! (összevonható)
4. Határozzuk meg a legnagyobb távolságot, ha van!

Specifikáció:

```
Be: n∈N, mag∈N[1..n]
Ki: van∈L, k∈N, v∈N
Fv: szigetkezdet, szigetvég...
Fv: keresvége:N->L x N,
    keresvége(i)=KERES(j=i..n, szigetvég(j))
Fv: táv:N->N, táv(i)=keresvége(i).ind - i
Ef: -
Uf: (van,k,)=FELTMAX(i=1..n, táv(i), szigetkezdet(i) és keresvége(i).van)
 és van -> v=keresvége(k).ind
```



Kis variáció…

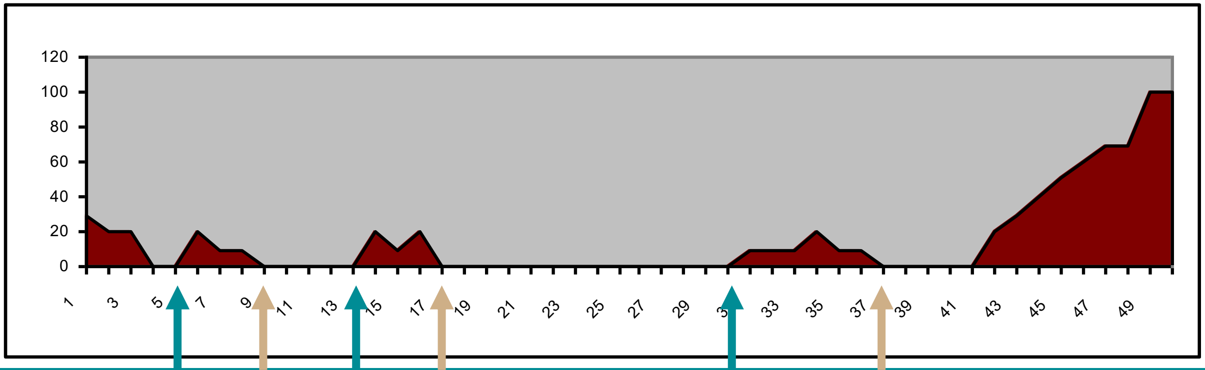
1. Válogassuk ki azokat a szigetkezdeteket, amelyeknek van vége is! (összevonható)
2. Másolással határozzuk meg hozzájuk a szigetvégeket! (összevonható)
3. Másolással határozzuk meg a távolságokat! (összevonható)
4. Határozzuk meg a legnagyobb távolságot, ha van!

Specifikáció:

```
Be: n∈N, mag∈N[1..n]
Ki: van∈L, k∈N, v∈N
Fv: szigetkezdet, szigetvég...
Fv: kereseleje:N->L x N,
    kereseleje(i)=KERES(j=-i..-1, szigetkezdet(-j))
Ef: -
Uf: (van,v,max)=FELTMAX(i=1..n, i - -kereseleje(i).ind,
                              szigetvég(i) és kereseleje(i).van) és
    van -> k=v-max+1
```

Hátulról keresés!

Hátulról keresés miatt a negált értéket kell visszaadnunk!



Visszavezetés

```
Fv: kereseleje:N->L x N,
    kereseleje(i)=KERES(j=-i..-1, szigetkezdet(-j))
Uf: (van,v,max)=MAX(i=1..n, i - -kereseleje(i).ind,
                            szigetvég(i) és kereseleje(i).van) és
    van -> k=v-max+1
```

Feltételes maximumkeresés		
maxind	~	v
maxért	~	max
e..u	~	1..n
f(i)	~	i+kereseleje(i).ind
T(i)	~	szigetvég(i) és kereseleje(i).van

Keresés (kereseleje)		
i	~	j
e..u	~	-i..-1
T(i)	~	szigetkezdet(-j)

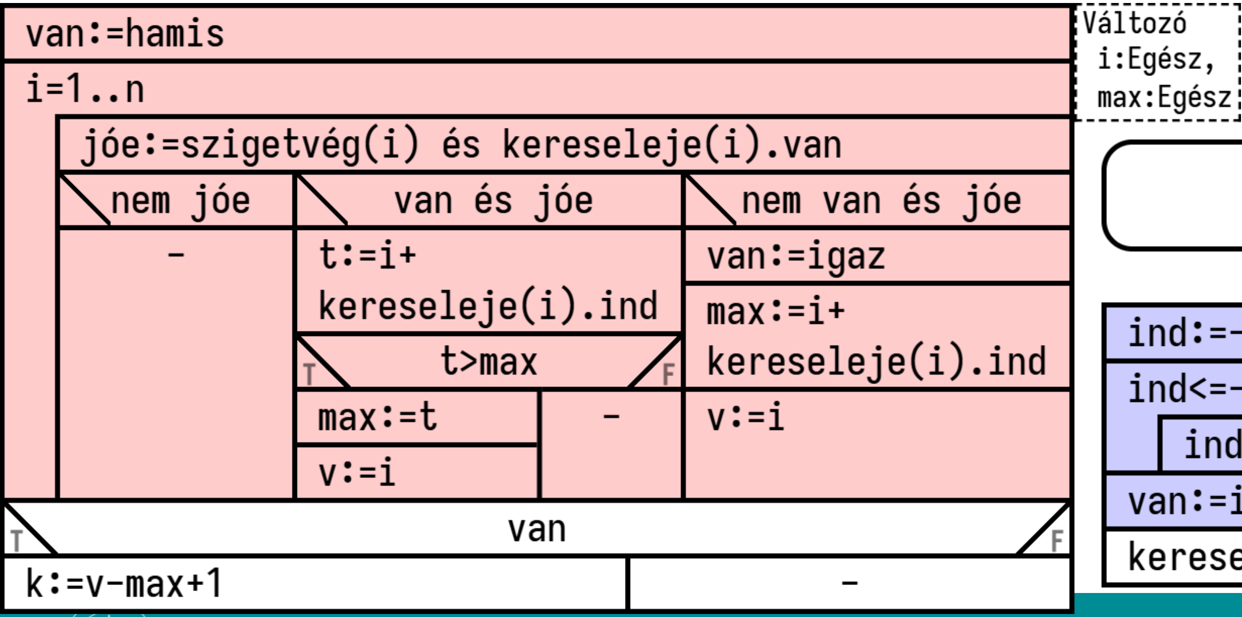
Algoritmus

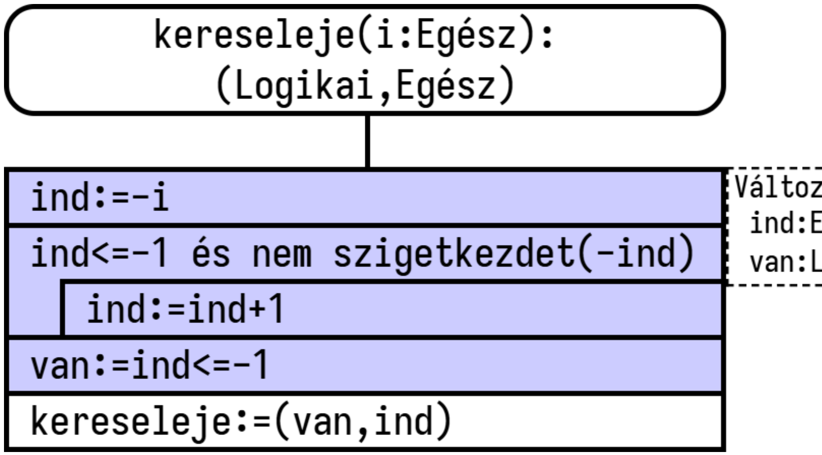
Feltételes maximumkeresés

```
maxind  ~  v
maxért  ~  max
e..u    ~  1..n
f(i)    ~  i+kereseleje(i).ind
T(i)    ~  szigetvég(i) és
           kereseleje(i).van
```

Keresés (kereseleje)

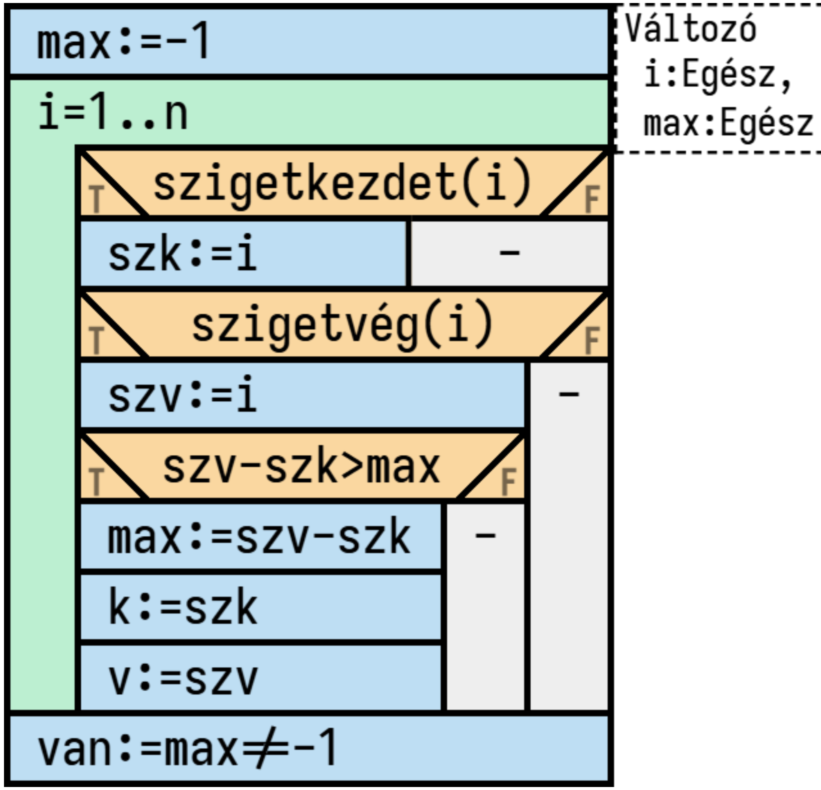
```
i     ~ j
e..u  ~ -i..-1
T(i)  ~ szigetkezdet(-j)
```

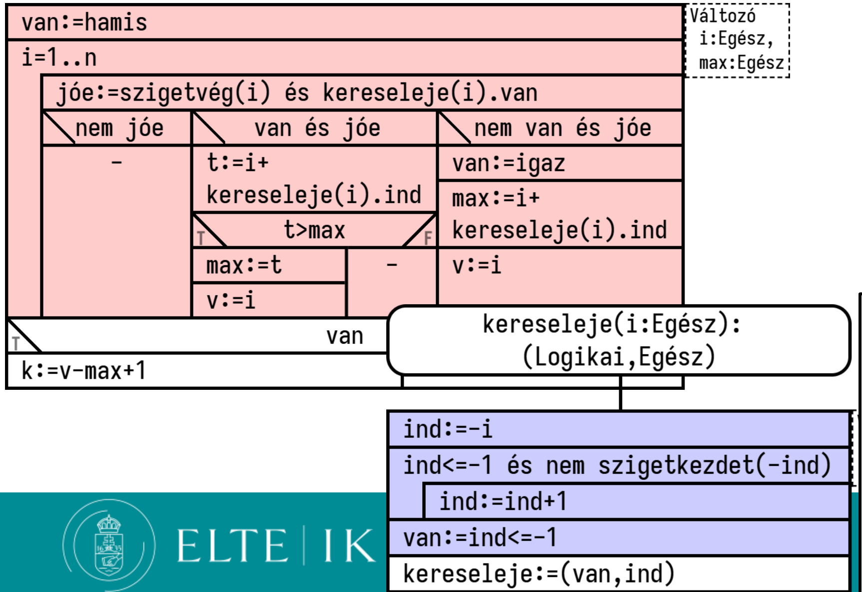


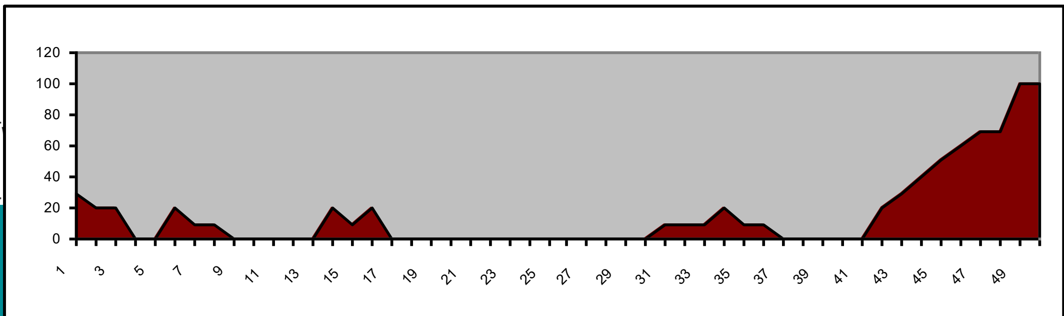


Algoritmikus gondolkodással

Ötlet: induljunk az elejétől, ha szigetkezdetet találunk, jegyezzük meg, ha véget, akkor számolhatjuk a sziget hosszát, és az eddigi maximummal összehasonlíthatjuk.

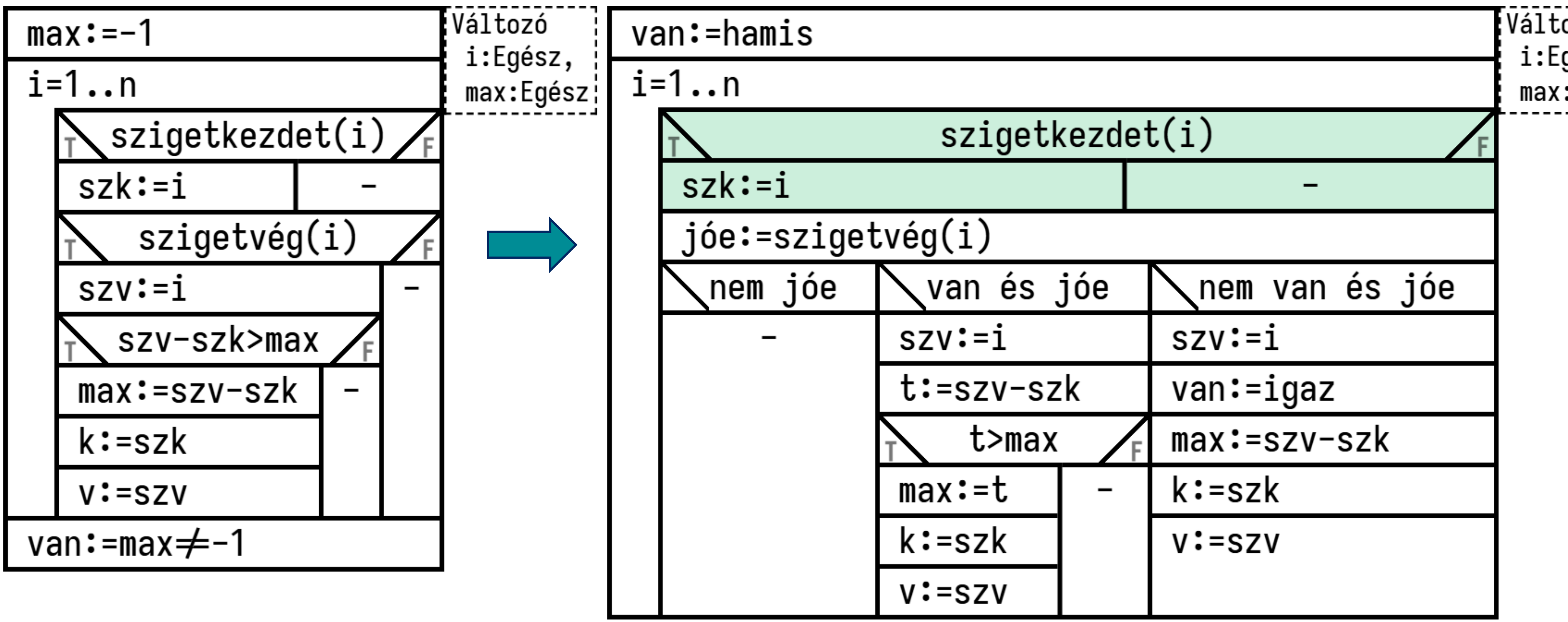


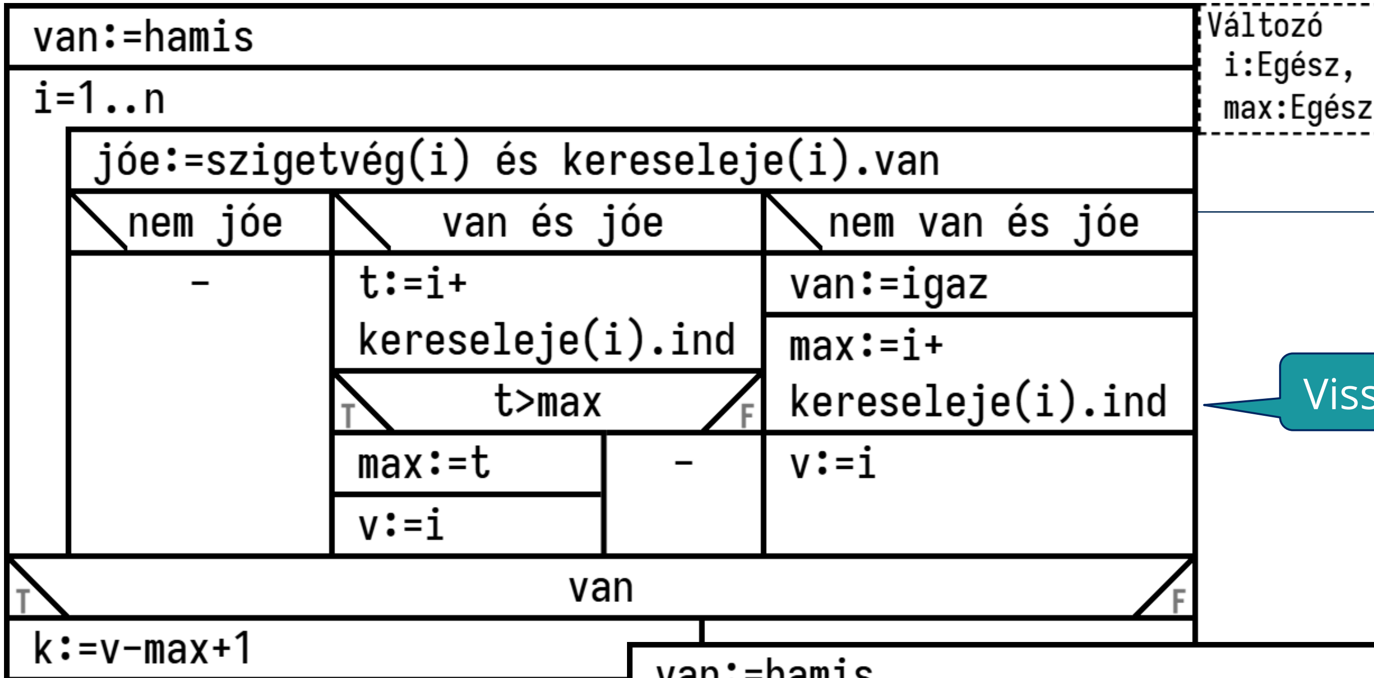




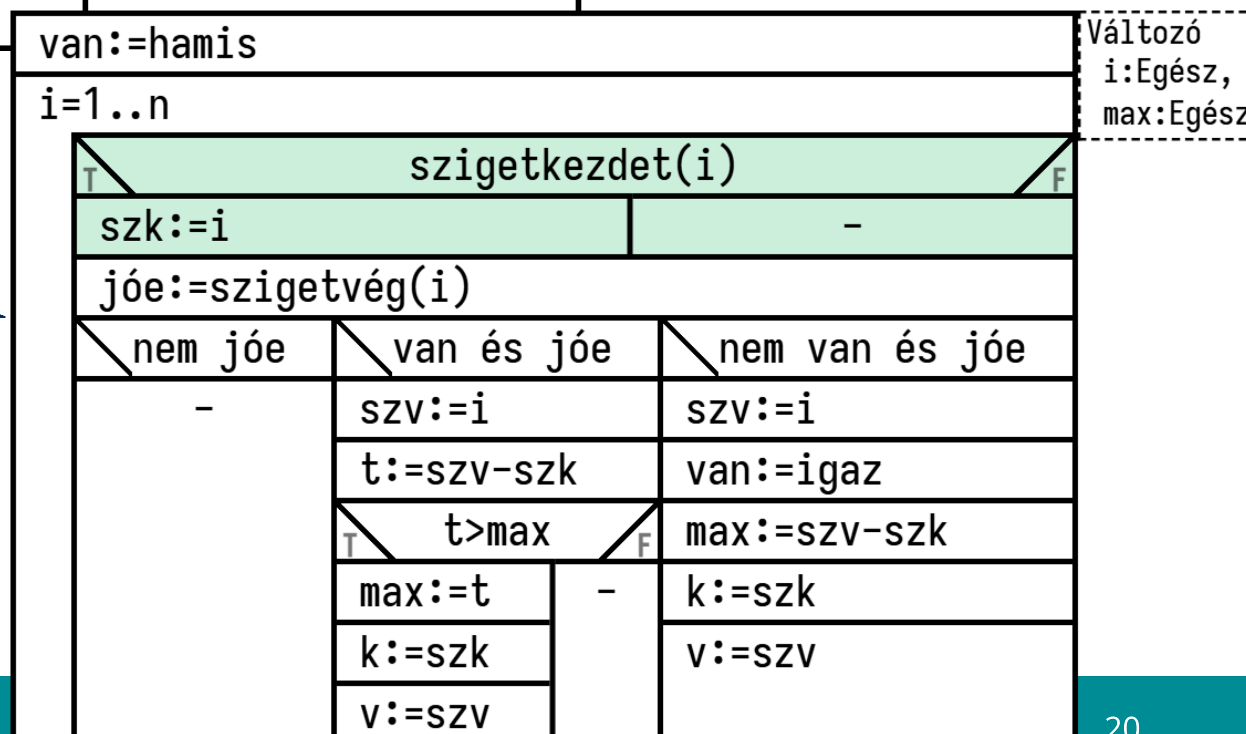
Programtranszformáció: közelítés

Mindkét megoldás feltételes maximumkeresés. Írjuk át a naív megoldásunkat struktúrájában ahhoz hasonlóvá!





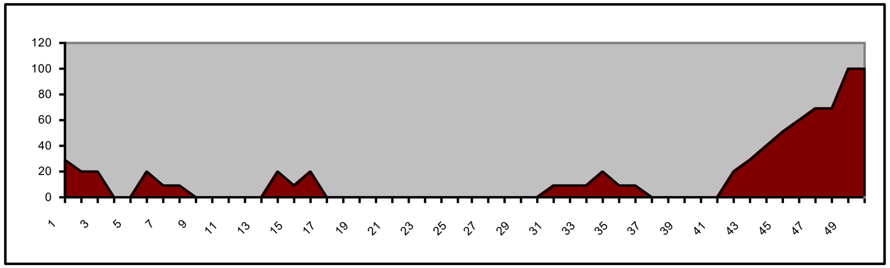
Visszavezetéssel



Algoritmikus gondolkodással

További közelítés: algoritmikus absztrakció – rekurzív függvény

Ötlet: Próbáljuk a kereseleje függvényt rekurzívan felírni! Ez a függvény minden pontban megmondja az adott ponthoz tartozó szigetkezdetet (ha van).



Specifikáció:

```
Fv: kereseleje:N->N,
    kereseleje(i)={i, ha szigetkezdet(i);
                   0, ha i < 1;
                   kereseleje(i-1) egyébként}
Uf: (van,v,max)=FELTMAX(i=1..n, i-kereseleje(i),
                        szigetvég(i) és kereseleje(i)>0) és
    van -> k=v-max+1
```

Rekurzív függvény

Egyetlen feltételes maximumkeresés

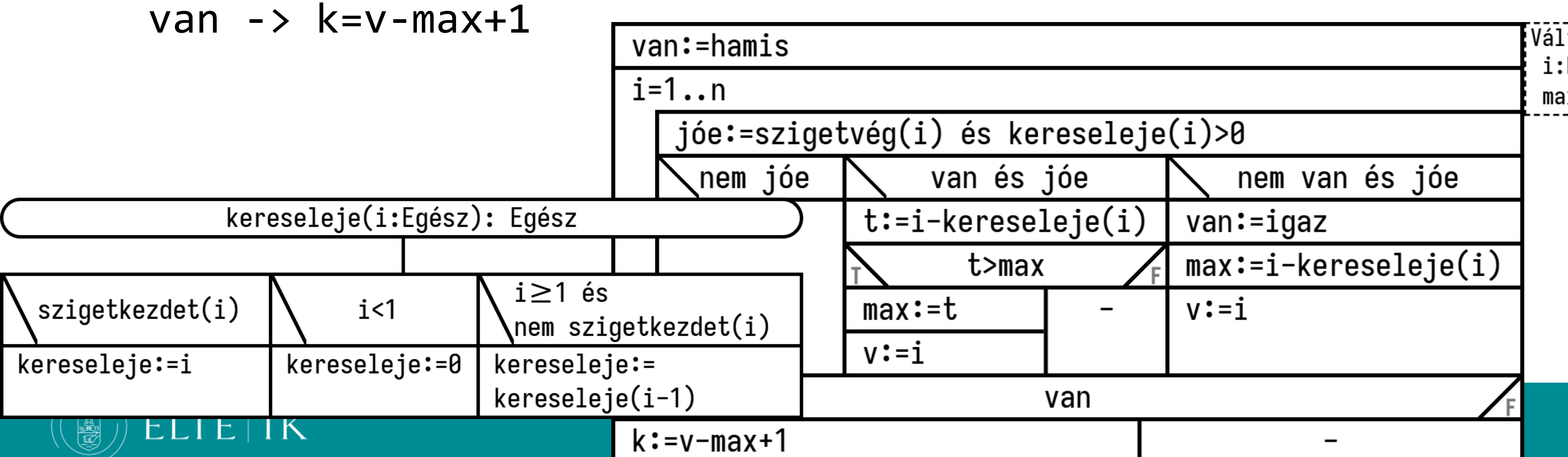
Visszavezetés

Feltételes maximumkeresés

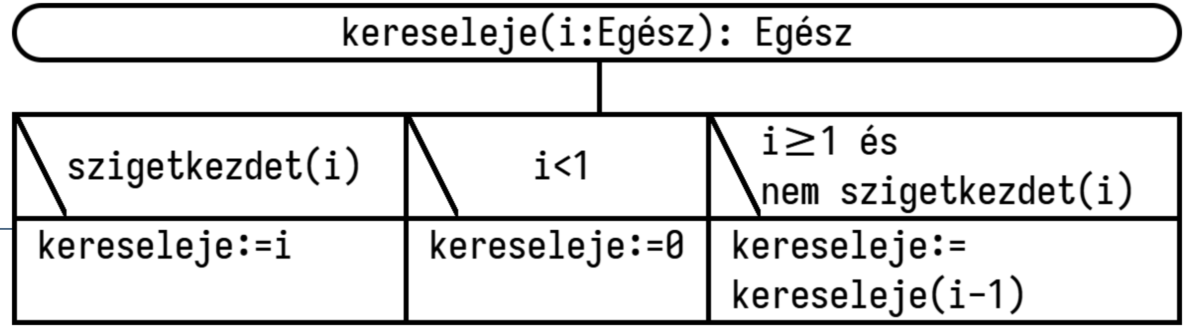
maxind	~	v
maxért	~	max
e..u	~	1..n
f(i)	~	i-kereseleje(i)
T(i)	~	szigetvég(i) és kereseleje(i)>0

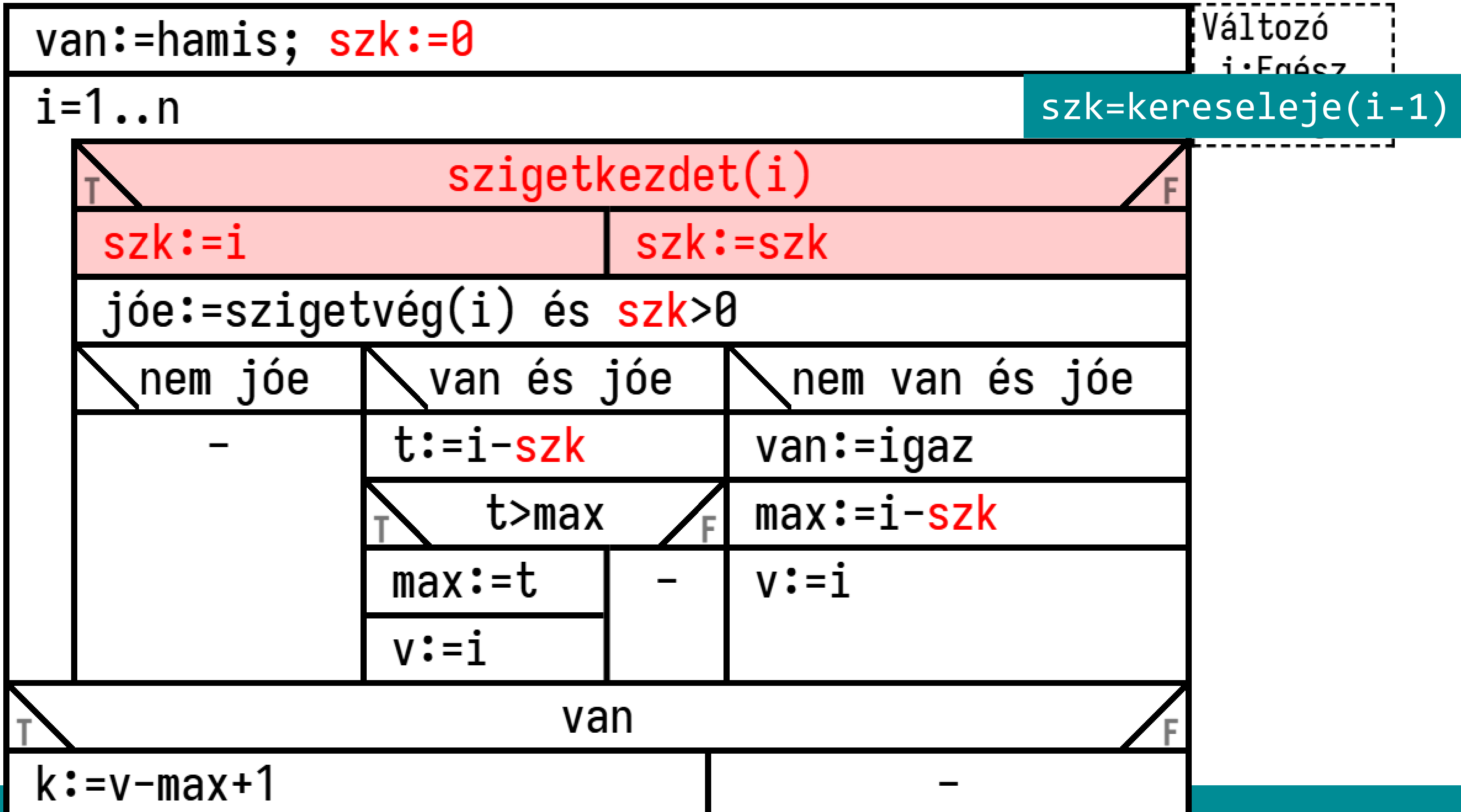
Specifikáció:

```
Fv: kereseleje:N->N,
    kereseleje(i)={i, ha szigetkezdet(i);
                   0, ha i < 1;
                   kereseleje(i-1) egyébként}
Uf: (van,v,max)=FELTMAX(i=1..n, i-kereseleje(i),
                        szigetvég(i) és kereseleje(i)>0) és
    van -> k=v-max+1
```



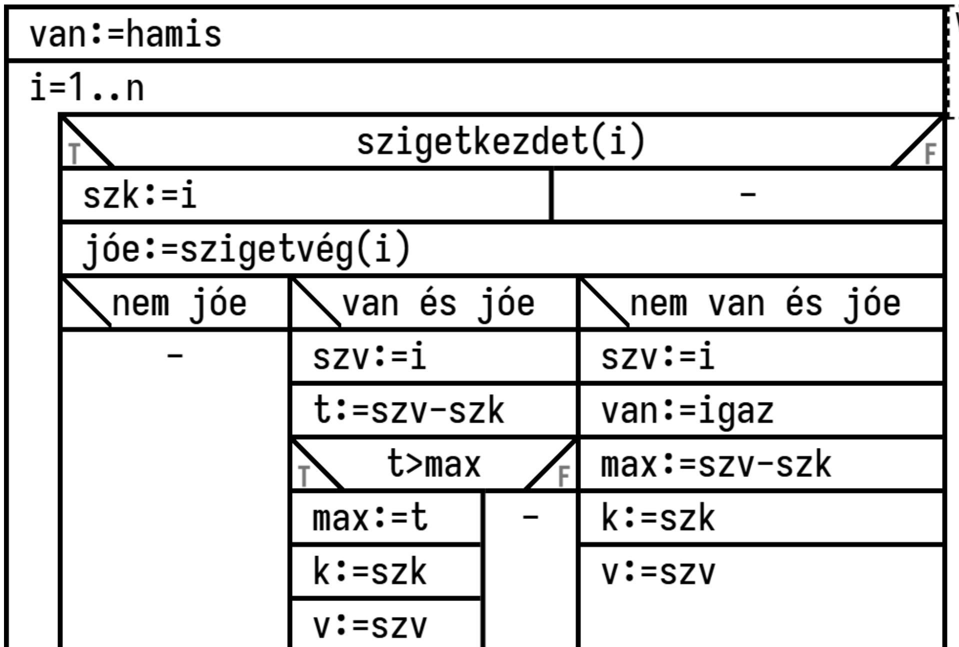
Rekurzió átírása



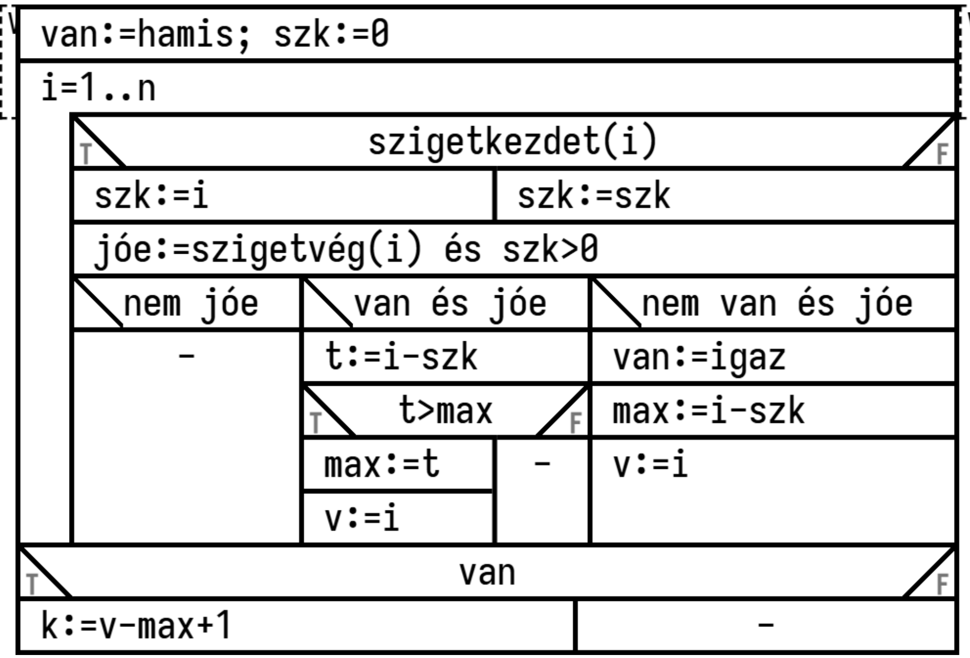


Összehasonlítás

Algoritmikus gondolkodással

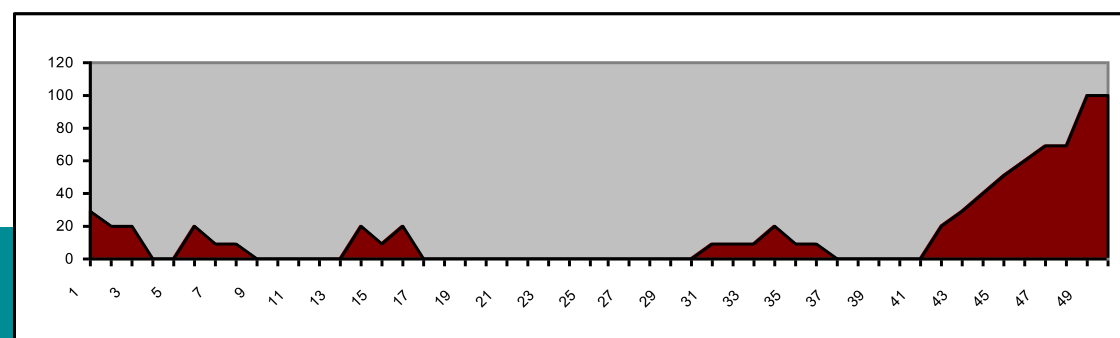


Rekurzív függvénnyel



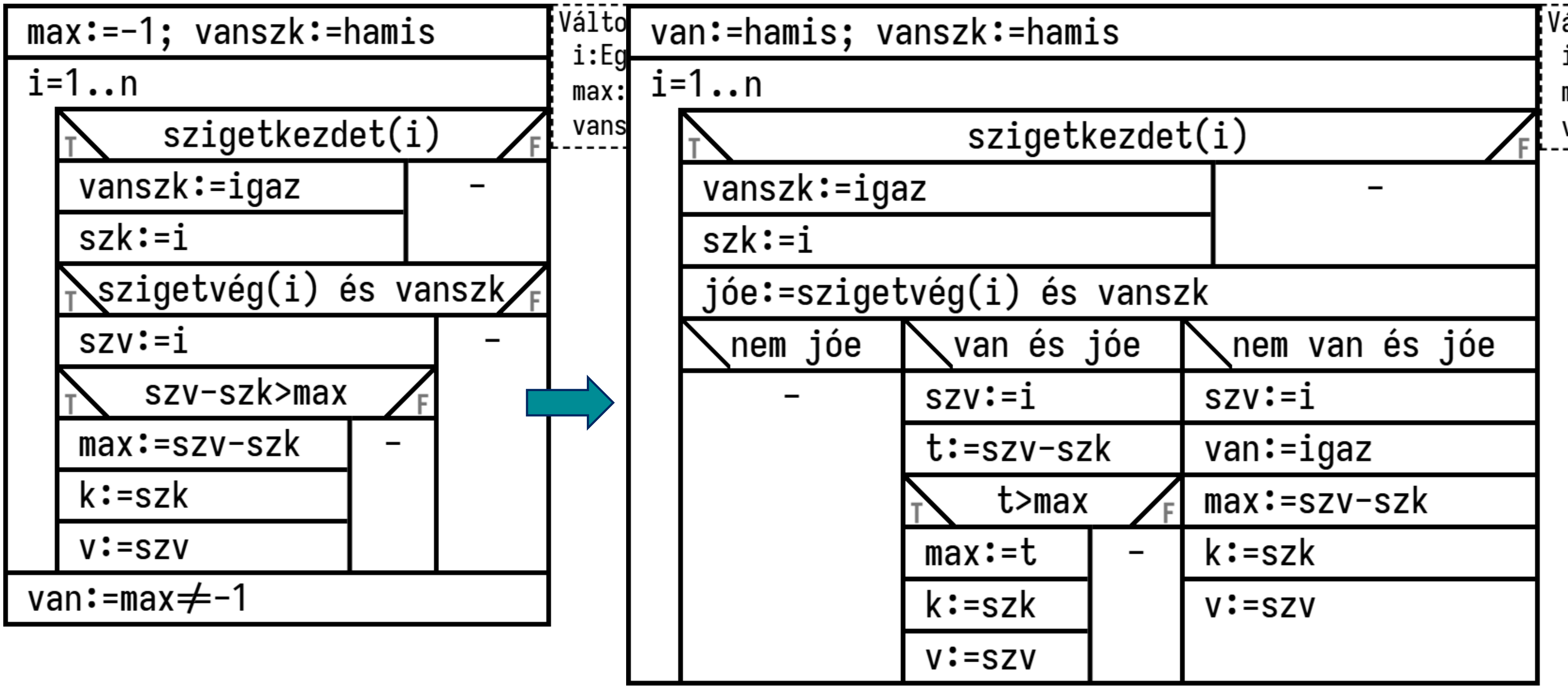
Funkcionálisan ugyanaz. Sőt!

Kiderült, hogy a bal oldali ROSSZ!



szk>0: nem veszi figyelembe, hogy korábban volt-e már szigetkezdet

Újra: algoritmikus gondolkodással



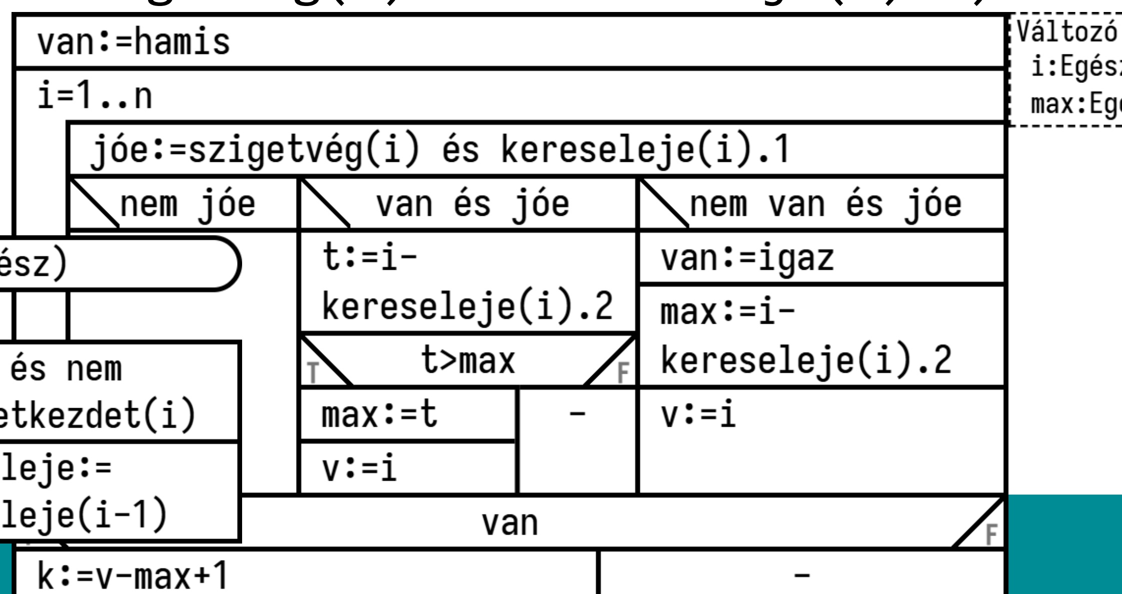
Újra: rekurzió és visszavezetés

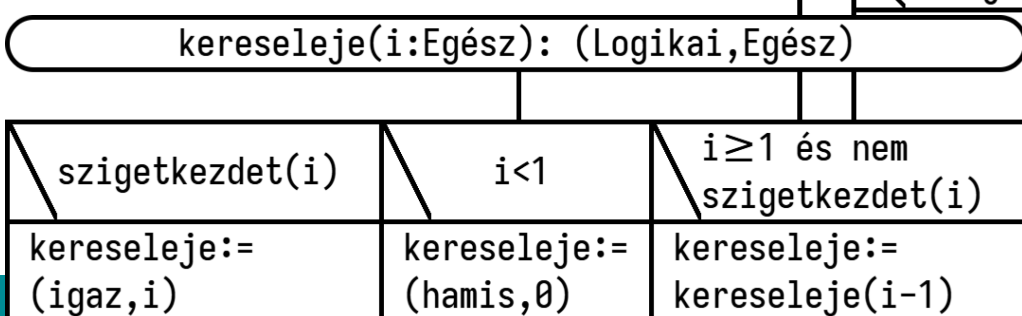
Feltételes maximumkeresés

```
maxind  ~  v
maxért  ~  max
e..u    ~  1..n
f(i)    ~  i-kereseleje(i).2
T(i)    ~  szigetvég(i) és
           kereseleje(i).1
```

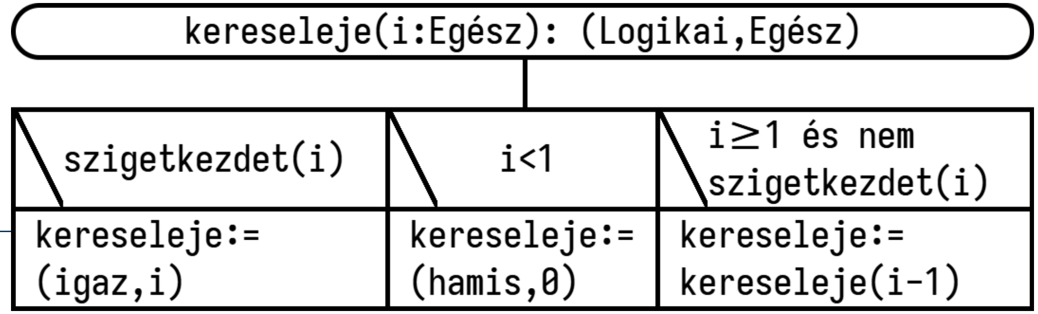
Specifikáció:

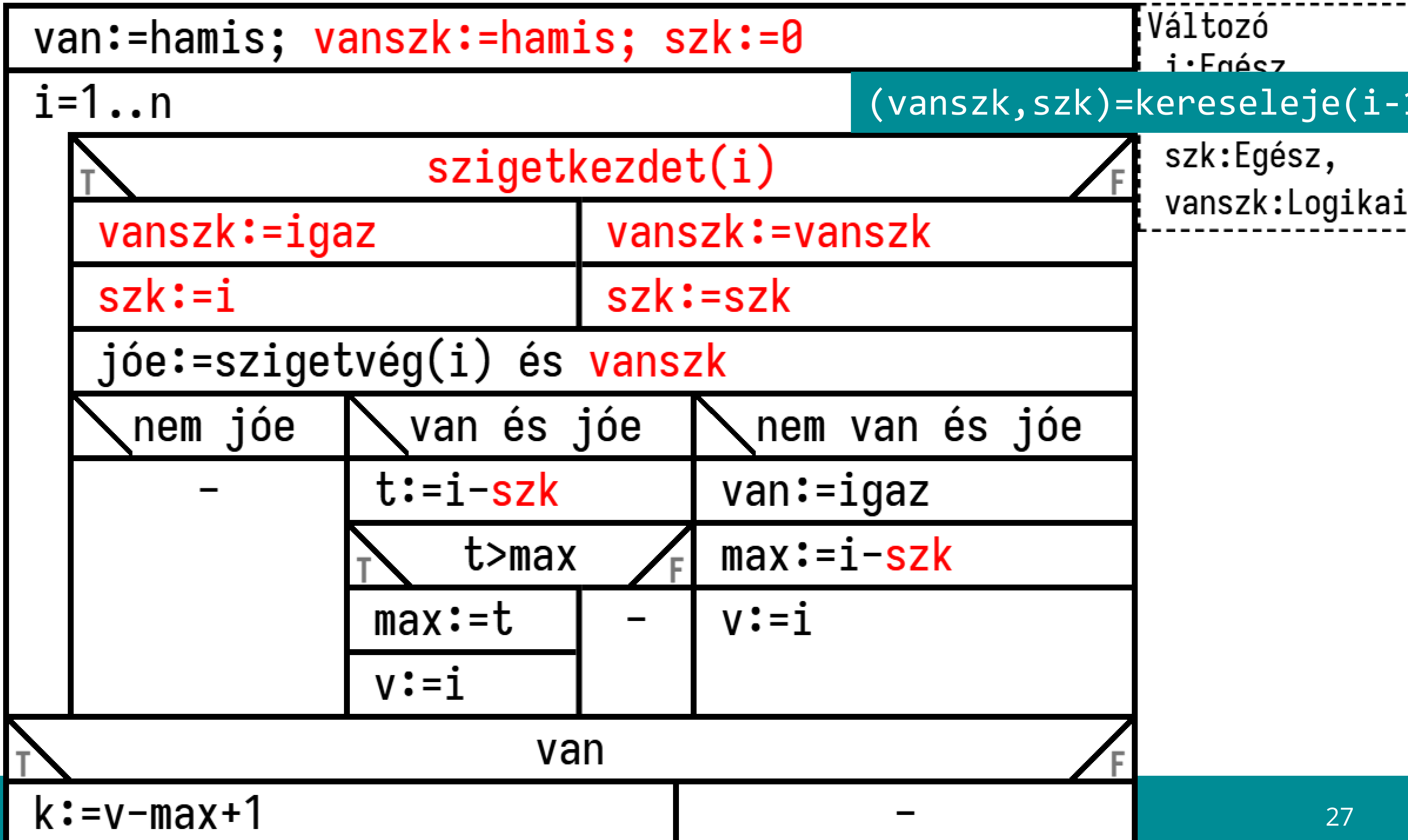
```
Fv: kereseleje:N->L x N,
    kereseleje(i)={(igaz, i), ha szigetkezdet(i);
                   (hamis,0), ha i < 1;
                   kereseleje(i-1) egyébként}
Uf: (van,v,max)=FELTMAX(i=1..n, i-kereseleje(i).2,
                        szigetvég(i) és kereseleje(i).1) és
    van -> k=v-max+1
```





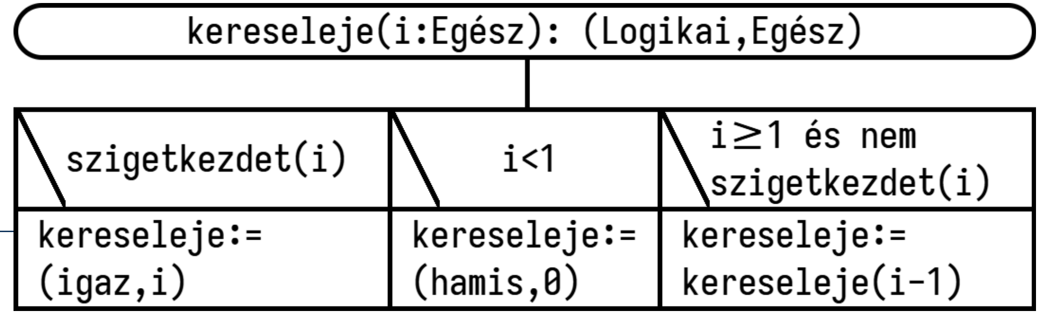
Újra: rekurzió átírása

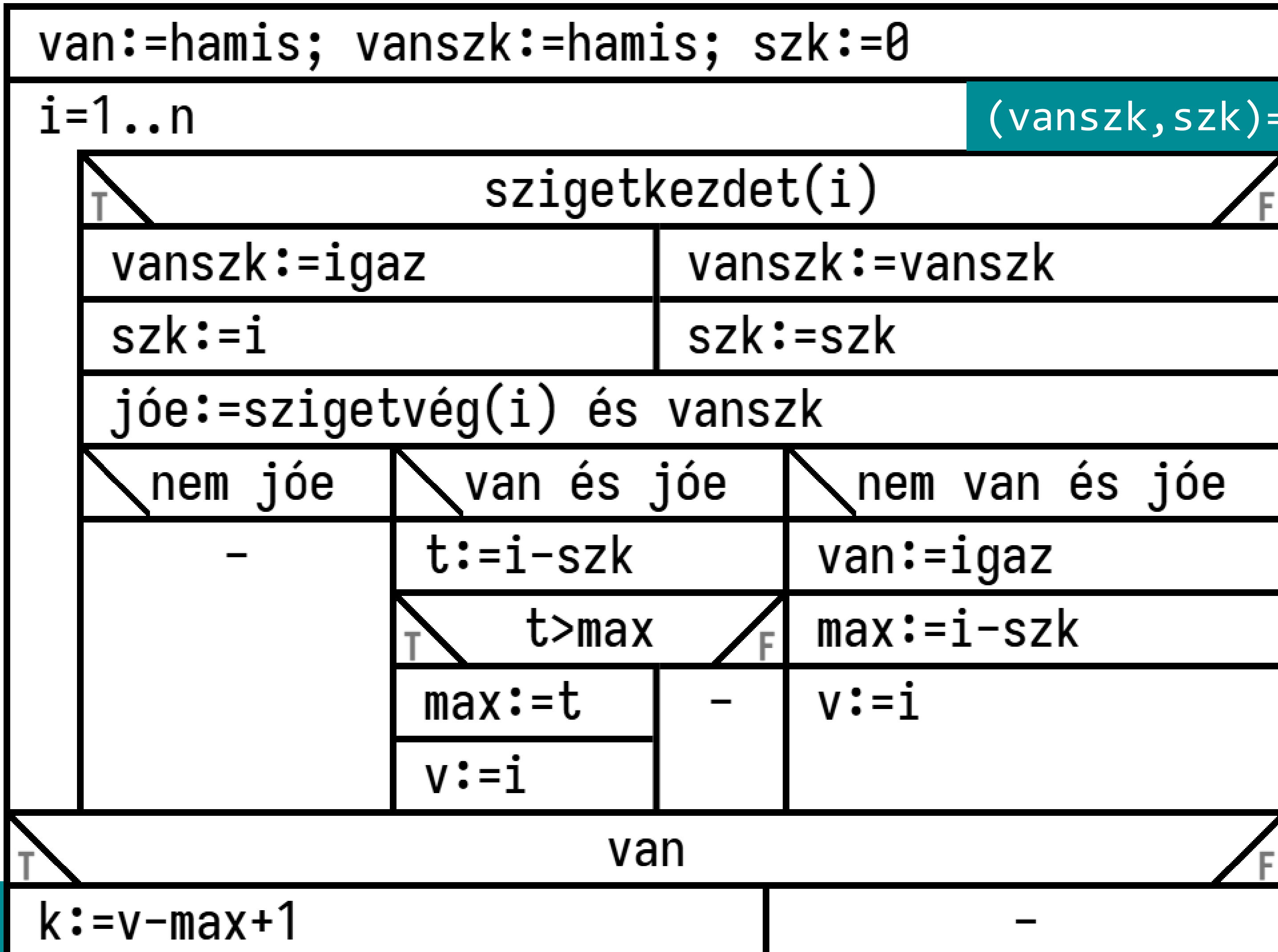




(vanszk,szk)=kereseleje(i-1)

Újra: rekurzió átírása





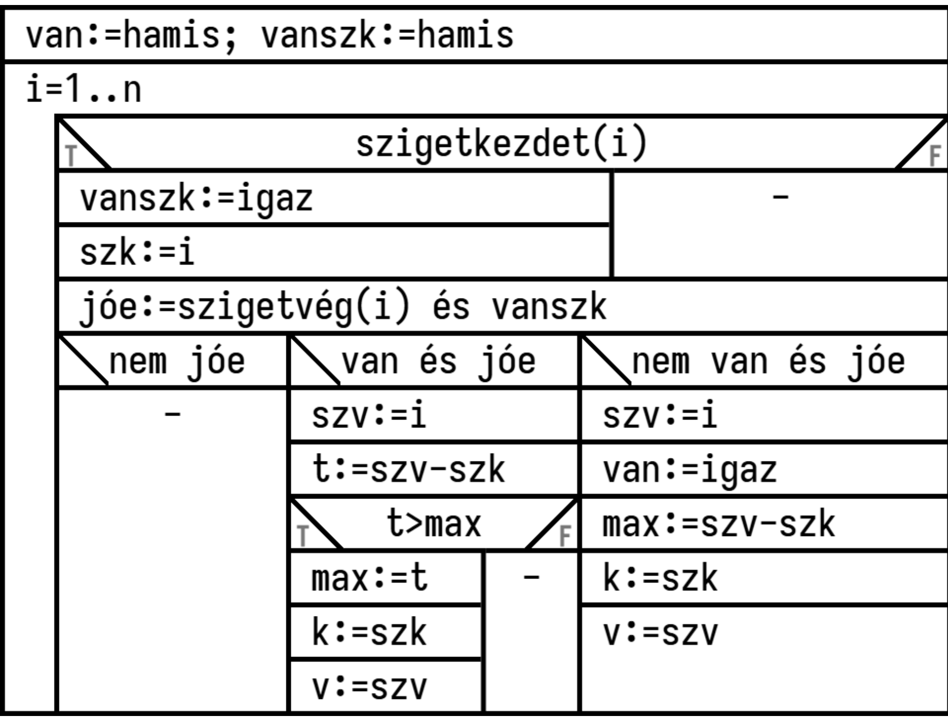
Változó
i:Egész

(vanszk,szk)=kereseleje(i-1)

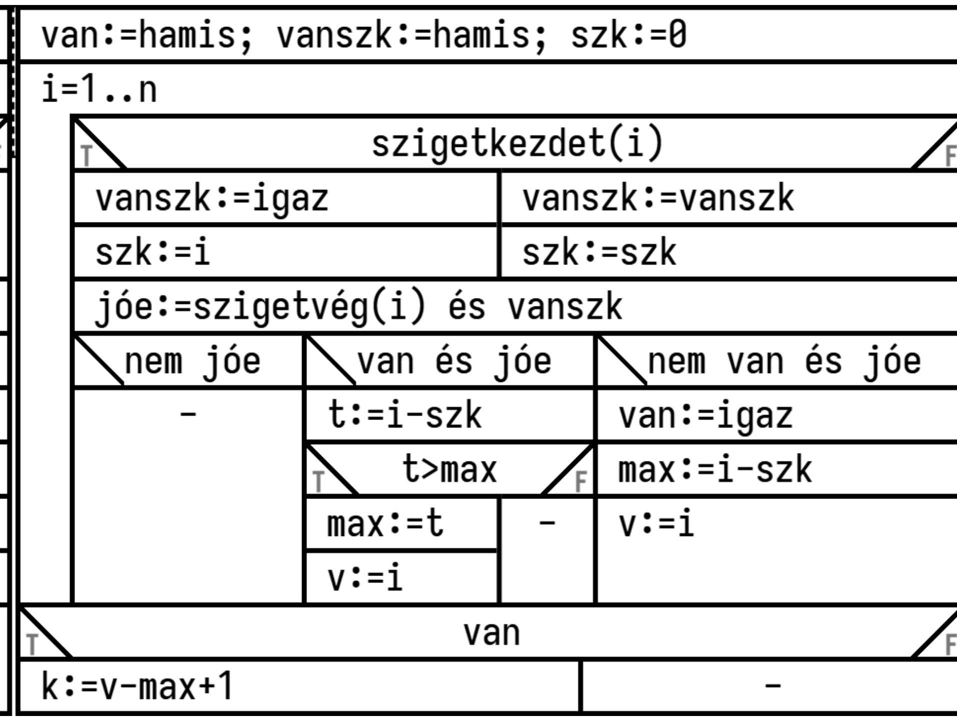
szk:Egész,
vanszk:Logikai

Újra: összehasonlítás

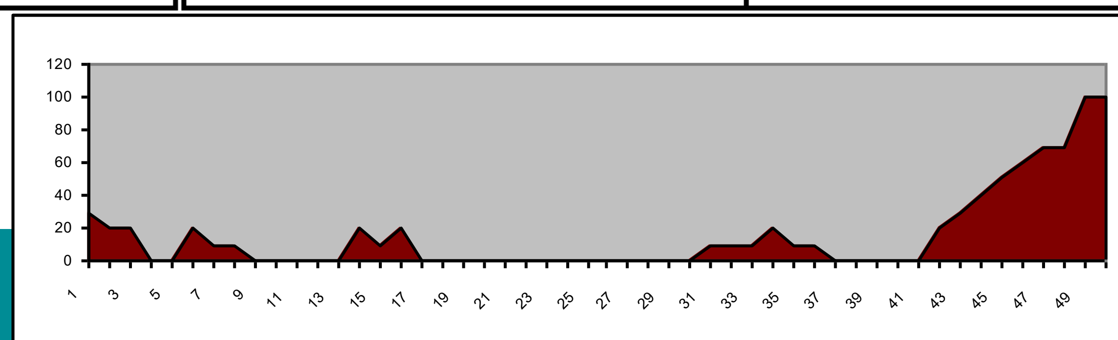
Algoritmikus gondolkodással



Rekurzív függvénnyel

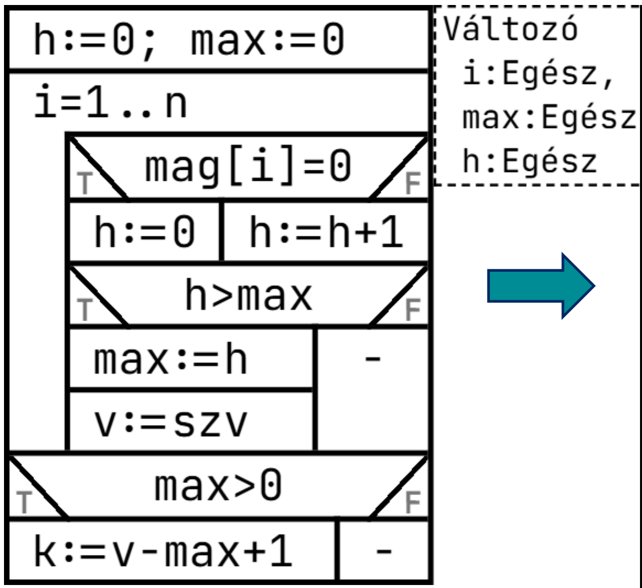


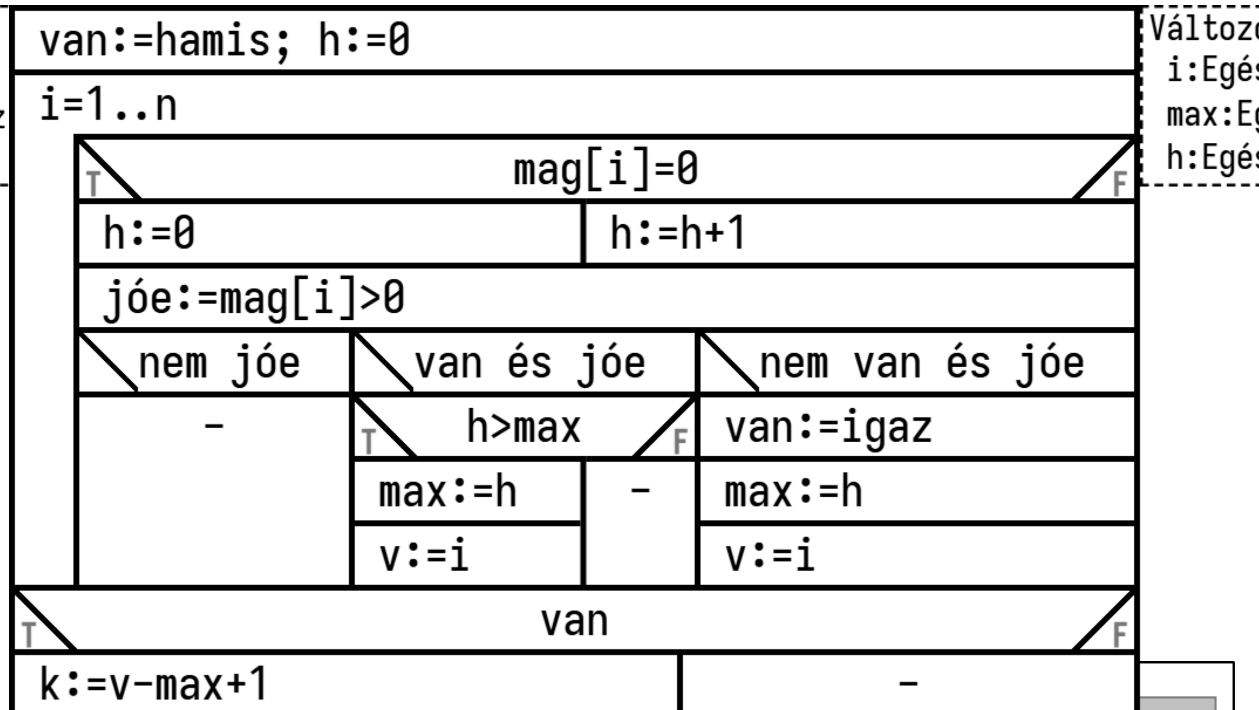
Funkcionálisan ugyanaz!

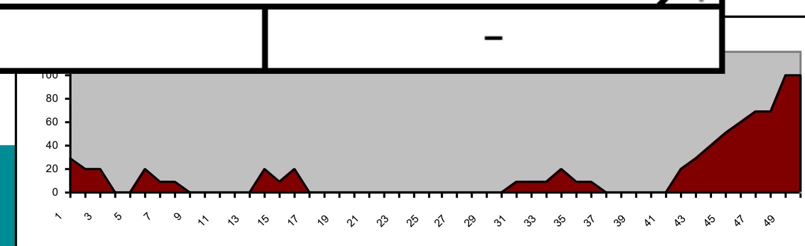


Másképp: algoritmikus gondolkodással

Ötlet: ha szárazföld fölött vagyunk, akkor növeljünk egy változót, és ennek a maximuma kell!







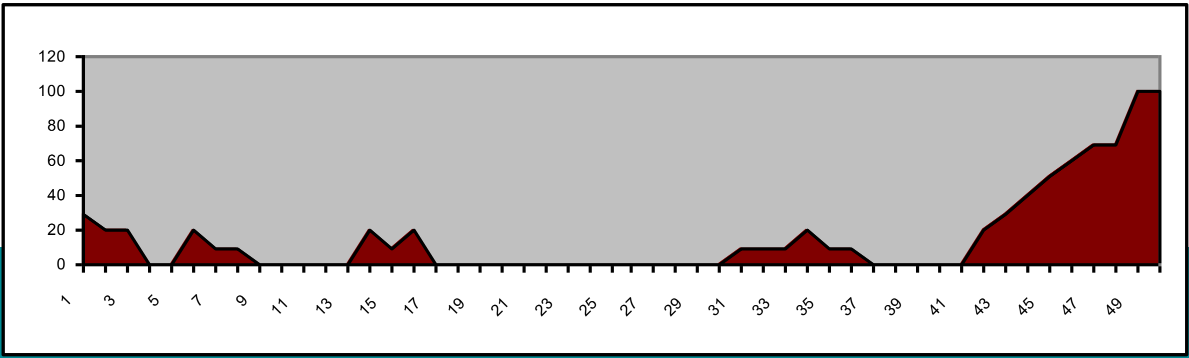
Másképp: rekurzív függvény

Ötlet: vezessünk be egy függvényt, amely minden pontban megmondja, hogy mekkora a távolság a szigetkezdet óta (tengernél 0). Hol veszi fel ez a legnagyobb értékét?

Specifikáció:

```
Fv: hossz:N->N, hossz(i)={0, ha i<1;
                          0, ha mag[i]=0;
                          hossz(i-1)+1 egyébként}
Uf: (van,v,max)=FELTMAX(i=1..n, hossz(i), mag[i]>0) és
    van -> k=v-max+1
```

Rekurzív függvény



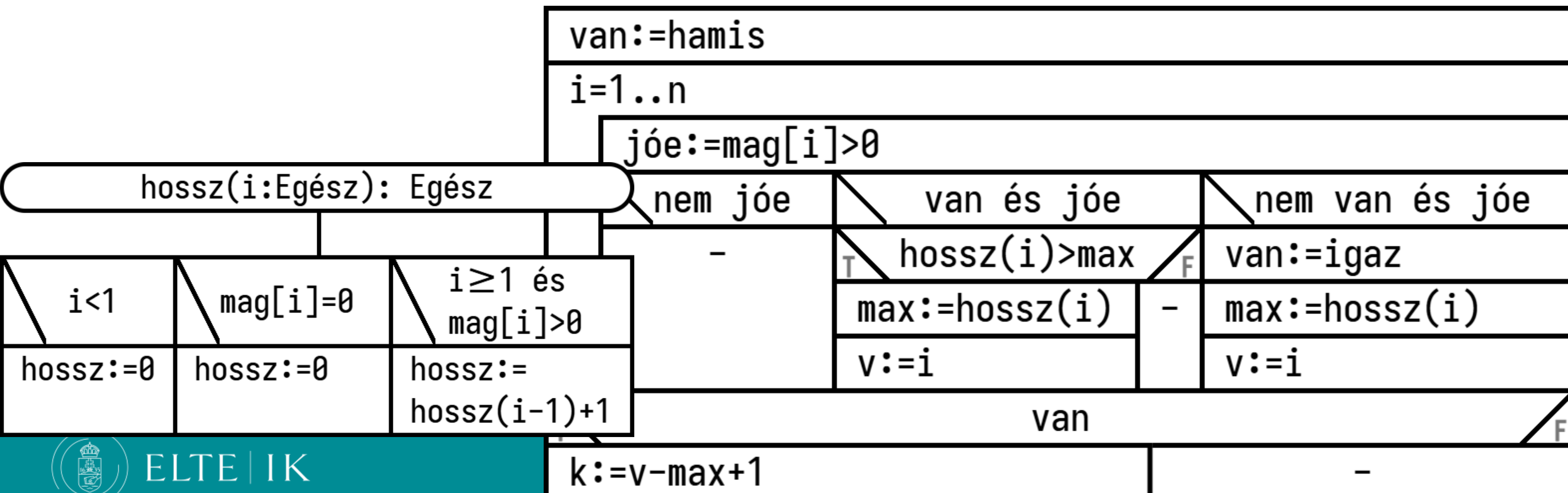
Másképp: visszavezetés

Feltételes maximumkeresés

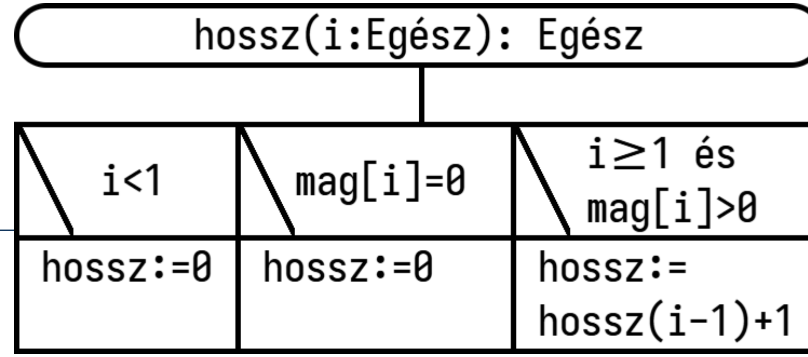
```
maxind  ~  v
maxért  ~  max
e..u    ~  1..n
f(i)    ~  hossz(i)
T(i)    ~  mag[i]>0
```

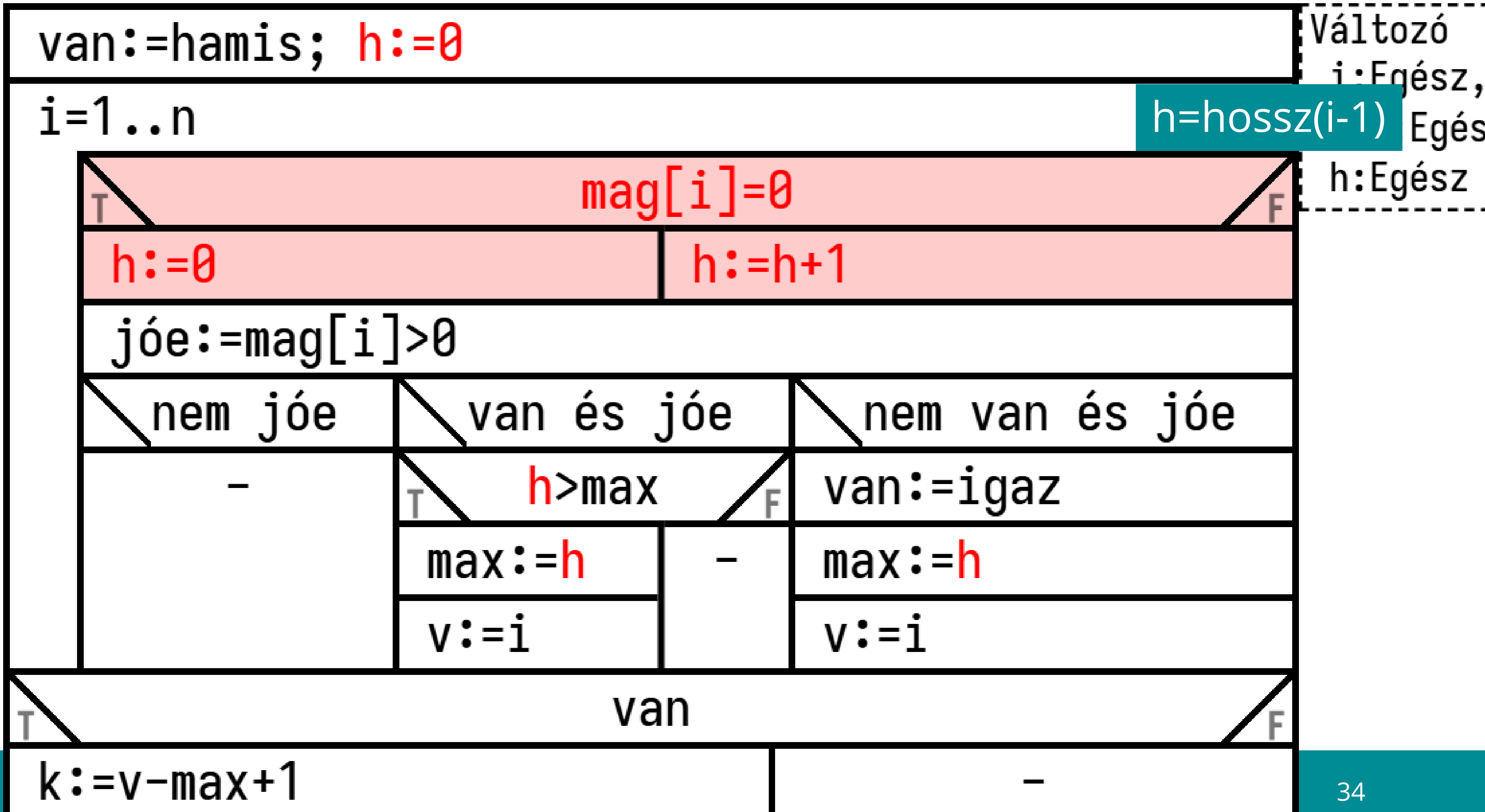
Specifikáció:

```
Fv: hossz:N->N, hossz(i)={0, ha i<1;
                          0, ha mag[i]=0;
                          hossz(i-1)+1 egyébként}
Uf: (van,v,max)=FELTMAX(i=1..n, hossz(i), mag[i]>0) és
    van -> k=v-max+1
```

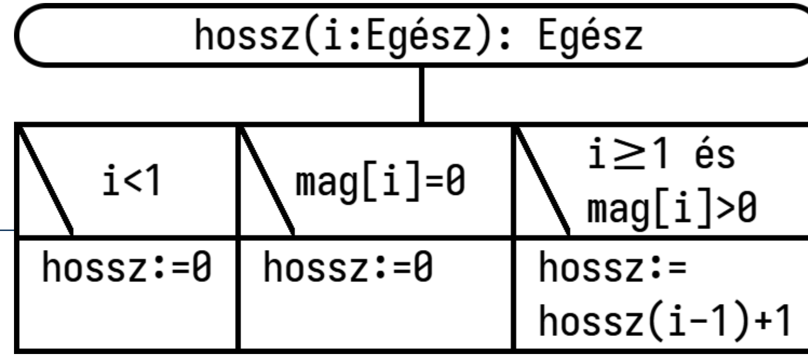


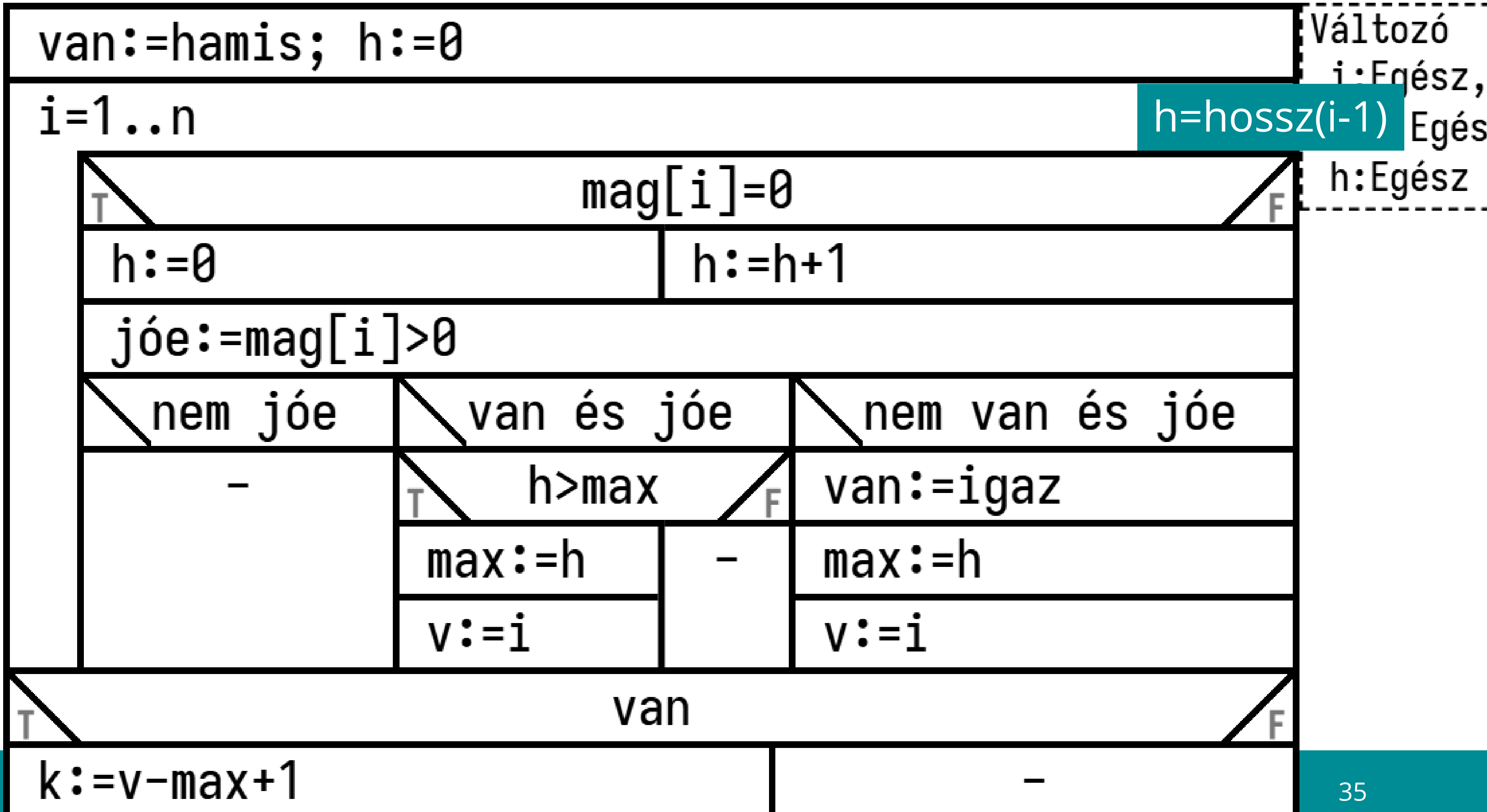
Másképp: rekurzió átírása





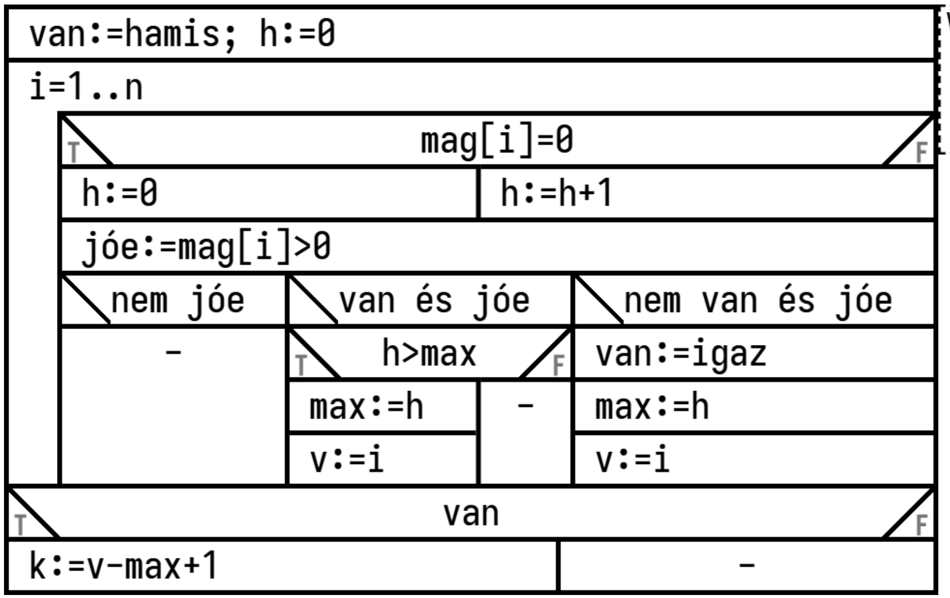
Másképp: rekurzió átírása



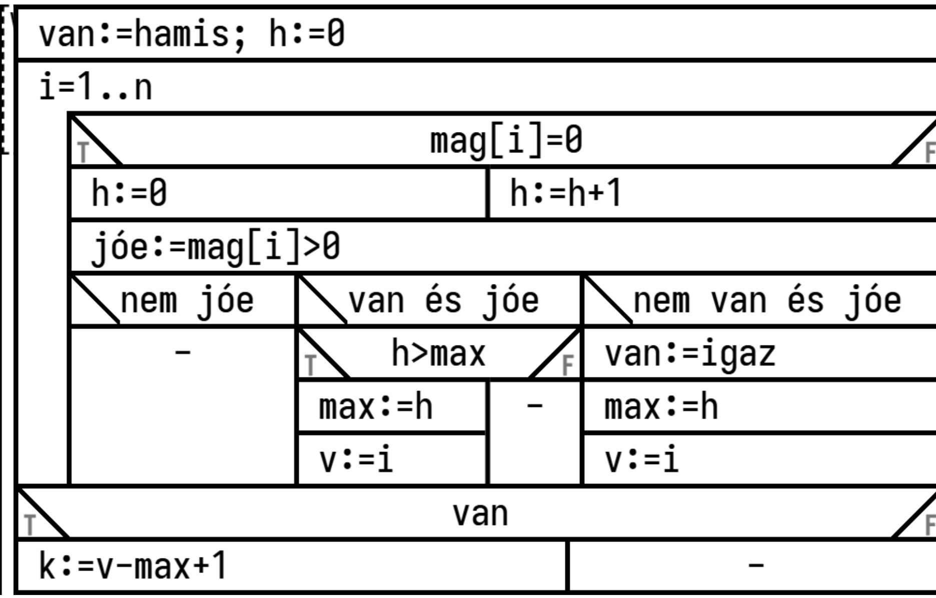


Újra: összehasonlítás

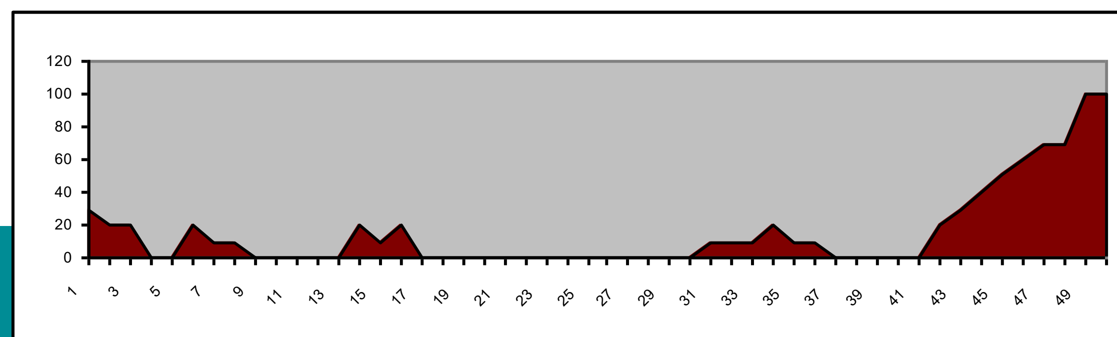
Algoritmikus gondolkodással



Rekurzív függvénnyel



Ez ugyanaz!!!



Rekurzív megoldás

Mindenhol, ahol „közben” megjegyzünk, gyűjtögetünk, és az előzővel ki tudjuk fejezni rekurzívan, ld. pl. az általánosított összegzést

Részösszegek

Feladat: Adott egy bolt bevétele egymást követő napokon. Melyik napon érte el vagy lépte át az addigi összbevétel a k Ft-ot?

```
Be: n∈N, ft∈N[1..n], k∈N
Ki: van∈L, nap∈N
```

k=20

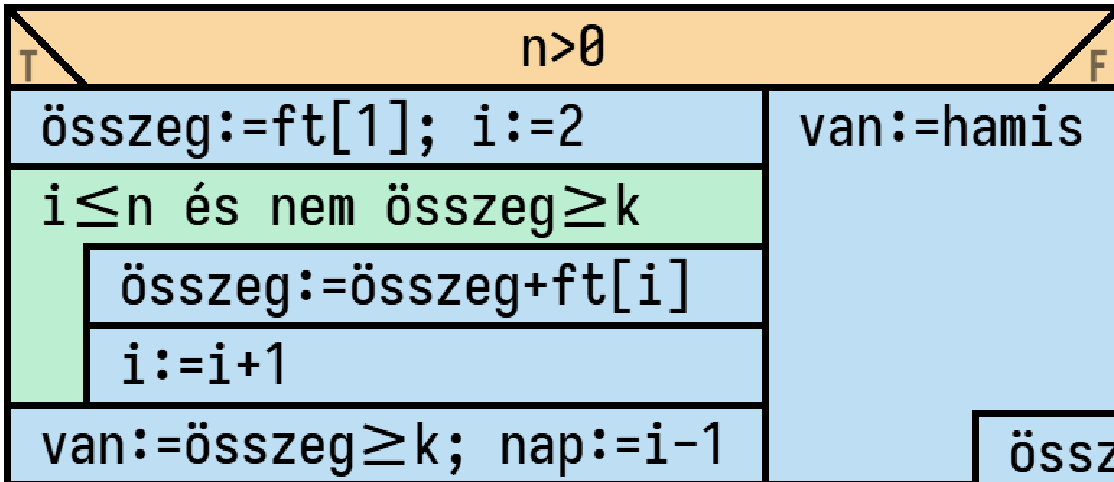
1	2	3	4	5	6	7	n=8
5	3	4	8	4	5	10	1

1	2	3	4	5	6	7	n=8
1	1	1	1	1	1	1	1

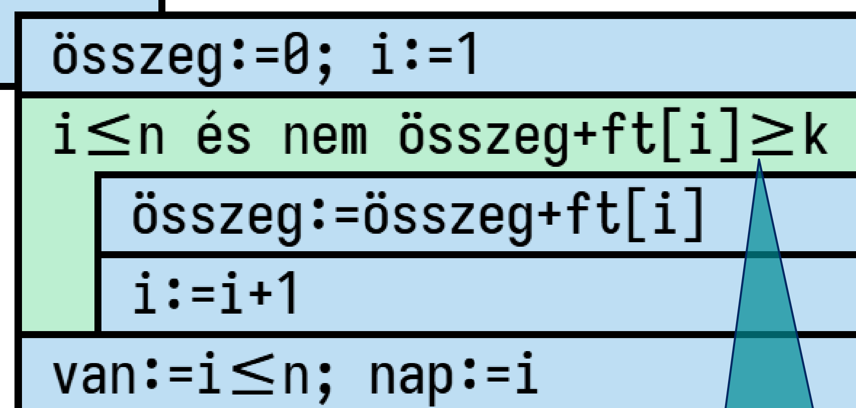
1	2	3	4	5	6	7	n=8
25	3	4	8	4	5	10	1

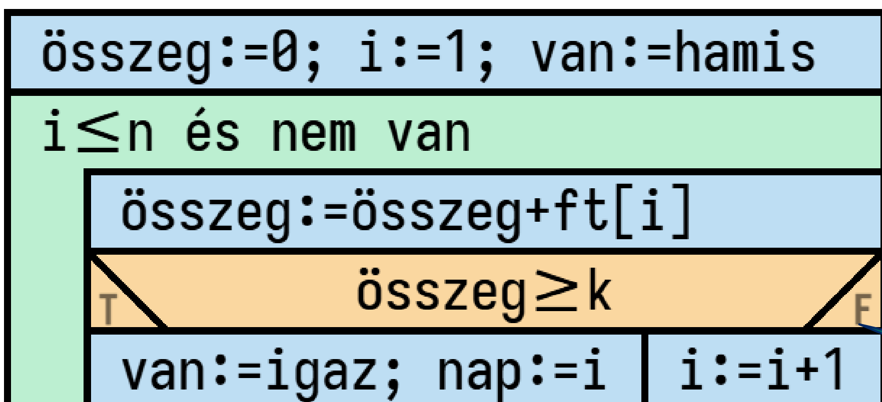
Részösszegek: algoritmikus gondolkodás

1	2	3	4	5	6	7	n=8
5	3	4	8	4	5	10	1



Erőltetett! Bonyolult.





Feltétel a ciklusmagban

Feltétel a ciklusfeltételben

1	2	3	4	5	6	7	n=8
5	3	4	8	4	5	10	1

Részösszegek

Adott egy bolt bevétele egymást követő napokon. Melyik napon érte el vagy lépte át az addigi összbevétel a k Ft-ot?

Specifikáció

```
Be: n∈N, ft∈N[1..n], k∈N
Ki: van∈L, nap∈N
Fv: összeg:N->N,
    összeg(i)=SZUMMA(j=1..i, ft[i])
Ef: -
Uf: (van,nap)=KERES(i=1..n, összeg(i)>=k)
```

Részösszegek: visszavezetés, algoritmus

1	2	3	4	5	6	7	n=8
5	3	4	8	4	5	10	1

```
Fv: összeg:N->N, összeg(i)=SZUMMA(j=1..i, ft[i])
Uf: (van,nap)=KERES(i=1..n, összeg(i)>=k)
```

Keresés

ind	~	nap
e..u	~	1..n
T(i)	~	összeg(i)>=k

```
nap:=1
nap≤n és nem(összeg(nap)≥k)
    nap:=nap+1
van:=nap≤n
```

Feltétel a ciklusfeltételben

Összegzés (összeg)

i	~	j
e..u	~	1..i
f(i)	~	ft[j]

```
összeg(i:Egész):Egész

s:=0
j=1..i
    s:=s+ft[j]
összeg:=s
```

Változó
j:Egész

Részösszegek: visszavezetés, algoritmus

1	2	3	4	5	6	7	n=8
5	3	4	8	4	5	10	1

```
Fv: összeg:N->N, összeg(i)=SZUMMA(j=1..i, ft[i])
Uf: (van,nap)=KERES(i=1..n, összeg(i)>=k)
```

Keresés

```
ind     ~  nap
e..u    ~  1..n
T(i)    ~  összeg(i)>=k
```

Összegzés (összeg)

```
i     ~ j
e..u  ~ 1..i
f(i)  ~ ft[j]
```

```
van:=hamis; nap:=1
nem van és nap≤n
    összeg(nap)≥k
    T: van:=igaz  |  F: nap:=nap+1
```

Feltétel a ciklusmagban

```
összeg(i:Egész):Egész
s:=0
j=1..i
    s:=s+ft[j]
összeg:=s
```

```
Változó
  j:Egész
```

1	2	3	4	5	6	7	n=8
5	3	4	8	4	5	10	1

Részösszegek: rekurzió

Adott egy bolt bevétele egymást követő napokon. Melyik napon lépte át az addigi összbevétel a k Ft-ot?

Specifikáció

```
Be: n∈N, ft∈N[1..n], k∈N
Ki: van∈L, nap∈N
Fv: összeg:N->N,
    összeg(i)={0, ha i<1;
               összeg(i-1)+ft[i] egyébként}
Ef: -
Uf: (van,nap)=KERES(i=1..n, összeg(i)>=k)
```

Rekurzív függvény

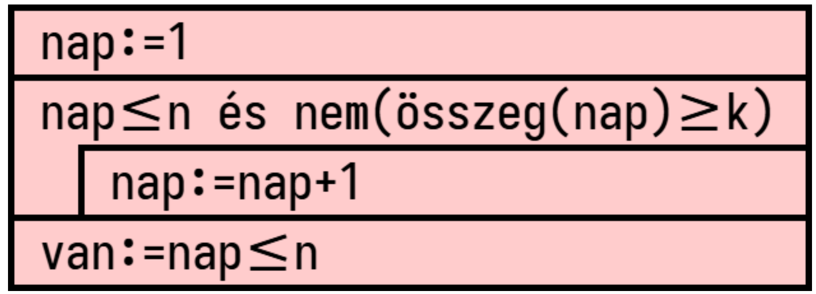
Részösszegek: visszavezetés, algoritmus

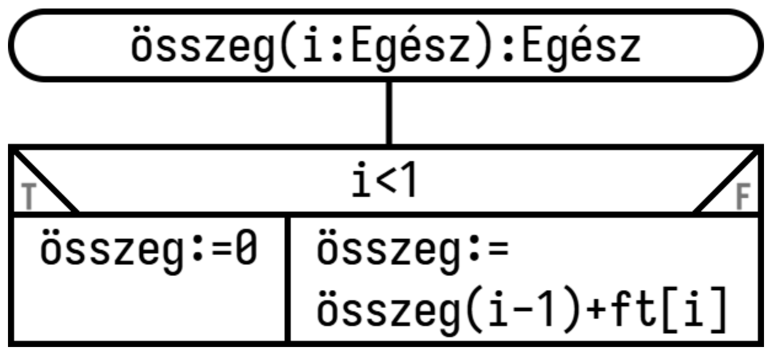
1	2	3	4	5	6	7	n=8
5	3	4	8	4	5	10	1

```
Fv: összeg:N->N, összeg(i)={0, ha i<1;
                             összeg(i-1)+ft[i] egyébként}
Uf: (van,nap)=KERES(i=1..n, összeg(i)>=k)
```

Keresés

ind	~	nap
e..u	~	1..n
T(i)	~	összeg(i)>=k





Részösszegek: rekurzió → iteráció

1	2	3	4	5	6	7	n=8
5	3	4	8	4	5	10	1

nap:=1
nap≤n és nem(összeg(nap)≥k)
nap:=nap+1
van:=nap≤n

összeg(i:Egész):Egész
i<1
T: összeg:=0
F: összeg:= összeg(i-1)+ft[i]

A kék rekurzívfüggvény-hívás helyére egy, az előzőtől függő kumulált értéket kell írni.

nap:=1; összeg:=0
nap≤n és nem(összeg+ft[nap]≥k)
összeg:=összeg+ft[nap]
nap:=nap+1
van:=nap≤n

összeg=összeg(i-1)

A ciklusba lépéskor, azaz a ciklusfeltételben már az új érték kell

összeg:=0; i:=1
i≤n és nem összeg+ft[i]≥k
összeg:=összeg+ft[i]
i:=i+1
van:=i≤n; nap:=i

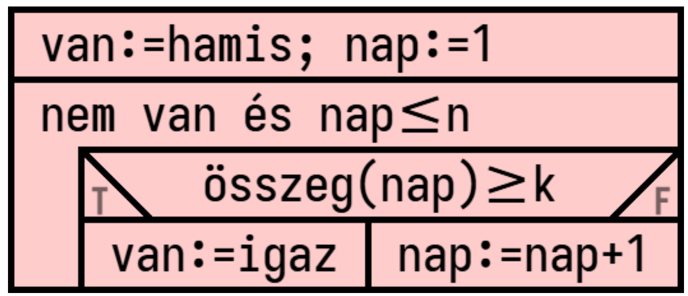
Részösszegek: visszavezetés, algoritmus

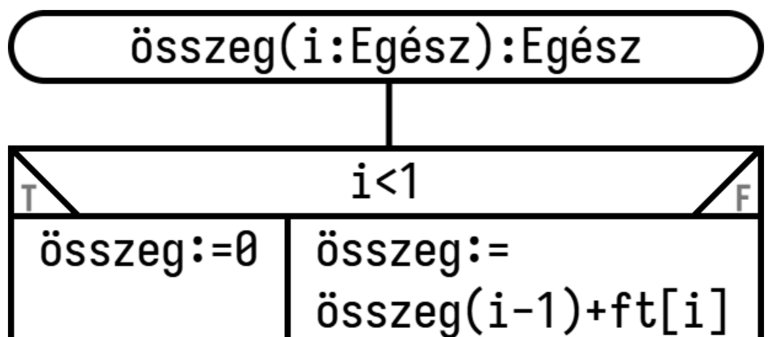
1	2	3	4	5	6	7	n=8
5	3	4	8	4	5	10	1

```
Fv: összeg:N->N, összeg(i)={0, ha i<1;
                            összeg(i-1)+ft[i] egyébként}
Uf: (van,nap)=KERES(i=1..n, összeg(i)>=k)
```

Keresés

ind	~	nap
e..u	~	1..n
T(i)	~	összeg(i)>=k





Részösszegek: rekurzió → iteráció

1	2	3	4	5	6	7	n=8
5	3	4	8	4	5	10	1

```
van:=hamis; nap:=1
nem van és nap≤n
    összeg(nap)≥k
    T: van:=igaz | F: nap:=nap+1
```

```
összeg(i:Egész):Egész
    i<1
    T: összeg:=0 | F: összeg:=összeg(i-1)+ft[i]
```

A kék rekurzívfüggvény-hívás helyére egy, az előzőtől függő kumulált értéket kell írni.

A ciklusba lépéskor

összeg=összeg(i-1)

```
van:=hamis; nap:=1; összeg:=0
nem van és nap≤n
    összeg:=összeg+ft[nap]
    összeg≥k
    T: van:=igaz | F: nap:=nap+1
```

```
összeg:=0; i:=1; van:=hamis
i≤n és nem van
    összeg:=összeg+ft[i]
    összeg≥k
    T: van:=igaz; nap:=i | F: i:=i+1
```